ESCUELA SUPERIOR DE INGENIERÍA

GRADO DE INGENIERÍA INFORMÁTICA

APLICACIÓN PARA ESTUDIAR ESTRATEGIAS DE GAMIFICACIÓN SOBRE LA PLATAFORMA MOODLE EN UN ENTORNO DE SIMULACIÓN CON ANYLOGIC

AUTOR: DANIEL GUERRERO MOLINA

Cádiz, octubre 2016

ESCUELA SUPERIOR DE INGENIERÍA

GRADO DE INGENIERÍA INFORMÁTICA

APLICACIÓN PARA ESTUDIAR ESTRATEGIAS DE GAMIFICACIÓN SOBRE LA PLATAFORMA MOODLE EN UN ENTORNO DE SIMULACIÓN CON ANYLOGIC

DIRECTOR: MERCEDES RUIZ CARREIRA
AUTOR: DANIEL GUERRERO MOLINA

Cádiz, octubre 2016

Dedicado a
mis padres.

Agradecimientos

Quiero dar las gracias a aquellas personas que me han apoyado no solo en la realización de este trabajo, sino durante toda la carrera y algunas, durante toda la vida.

A mi familia y en especial a mis padres, que gracias a su esfuerzo he podido tener la oportunidad de estudiar esta carrera y, porque todo lo que soy se lo debo a ellos. A mi novia, por estar siempre a mi lado, porque seguimos superando etapas de la vida, pero siempre juntos.

Agradecerle a Mercedes, mi tutora, su paciencia y dedicación. Por todos esos consejos, por demostrar que para ella lo primero son sus alumnos, por esas charlas de tutoría, en las que parecía más mi psicóloga que mi profesora. Si algún día llego a ser un buen profesional, ella tendrá gran parte de culpa.

Por último, acordarme de aquellos compañeros que se han convertido en amigos y, con los que he pasado de ser compañeros de universidad a ser compañeros de trabajo, por haber estado ahí en todo momento y por aguantarme tantos años.

Muchas gracias a todos.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

En este documento se redactarán todos los aspectos claves del desarrollo de la aplicación Strategy Simulator, aplicación desarrollada como trabajo fin de grado, del grado de ingeniería informática. Esta aplicación une dos conceptos innovadores, y en pleno crecimiento. Estos dos conceptos son la gamificación y la simulación, siendo esta última el pilar fundamental de este proyecto. A continuación se describirá la motivación del proyecto, los objetivos que se quieren alcanzar y los acrónimos utilizados en el documento.

1.1. Motivación

La aparición de la web 2.0 ha acelerado la creación de comunidades en torno a todo tipo de redes sociales, medios digitales, etc. Pero no siempre es fácil estimular la actividad dinámica y frecuente entre los miembros de una comunidad. Y aparece el concepto de Gamificación, que se podría definir como el empleo de mecánicas de juego en entornos y aplicaciones no lúdicas con el fin de potenciar la motivación, la concentración, el esfuerzo, la fidelización y otros valores positivos comunes a todos los juegos. Se trata de una nueva y poderosa estrategia para influir y motivar a grupos de personas.

La principal motivación de este proyecto radica en aplicar la gamificación a la plataforma de aprendizaje Moodle y así promover una mayor participación de sus usuarios. Dado que no se sabe si el impacto de aplicar la gamificación a Moodle sería positivo o negativo, surge el desarrollo de esta aplicación.

Esta aplicación se encarga de generar un modelo de simulación de cuál sería el comportamiento de los alumnos frente a una estrategia de gamificación aplicada a un curso de Moodle. Este modelo se ejecutaría en el software Anylogic. Esta aplicación también ofrece la posibilidad de aplicar una estrategia de gamificación a un curso real de Moodle y así ver cómo habrían sido los resultados. Además, estos resultados se pueden publicar en SmartWeb, una red social creada por un compañero de la UCA, así los alumnos podrán compartir sus logros con otros compañeros.

1.2. Objetivos y ámbito del sistema

El principal objetivo del sistema es ofrecer al usuario la posibilidad aplicar a los cursos de la plataforma Moodle estrategias de gamificación en un entorno de Simulación. Las estrategias de gamificación son diseñadas en una aplicación externa a ésta. Una estrategia de gamificación es una secuencia de actividades, en la que cada actividad tiene tiene una o varias recompensas que serán asignadas al usuario que supera la actividad con éxito. Es importante destacar que las actividades de la estrategia tienen unos o varios criterios que son los que permiten evaluar si una actividad se ha superado o no. La estrategia o estrategias están almacenadas en un archivo en formato XML, este archivo es leído por nuestra aplicación para cargar las estrategias y poder trabajar con ellas. En el Capítulo 7, se explicará de manera detallada la estructura del XML.

Esto permite realizar un estudio de las repercusiones, tanto positivas como negativas, que tendría el uso de gamificación en esta plataforma de aprendizaje. Al estar en un entorno de simulación permite al usuario obtener una muestra de cómo responderían los estudiantes ante esta propuesta. En este caso, la aplicación utilizada para ejecutar la simulación será Anylogic.
A parte de la simulación, el sistema ofrece la posibilidad de aplicar las estrategias a Moodle, obteniendo resultados reales de su base de datos y aplicando la estrategia a estos datos para ver los resultados que habrían obtenido los alumnos. Estos resultado se podrán guardar en una base de datos de otra aplicación independiente de esta, para un uso posterior.
Este sistema tiene una principal ventaja y es que gracias a la simulación podemos realizar un estudio de viabilidad y, en caso de obtener resultados negativos, se ahorraría el coste significativo que implicaría implantar este sistema en un entorno real.
El ámbito del sistema es que la aplicación pueda ser utilizada por cualquier persona, sin necesidad de que conozca conceptos de simulación. Está dirigida a docentes que hagan uso de la plataforma Moodle y que estén interesados en estudiar la posibilidad de gamificar el entorno de aprendizaje, señalando que en la actualidad es un método que está en crecimiento por los buenos resultados que ofrece.

1.2.1. Objetivos del sistema

En esta sección se describirán los principales objetivos del sistema. El objetivo principal es simular estrategias de gamificación aplicadas a Moodle en Anylogic, también ofrecer al usuario la posibilidad de aplicar estar estrategias a un curso real de Moodle, para posteriormente publicar los resultados obtenidos en la red social SmartWeb.
A continuación, en los Cuadros 1.1, 1.2, 1.3 y 1.4 se describen con mayor detalle todos los objetivos.

Código	OBJ-001
Descripción	Simular estrategias de gamificación sobre cursos de Moodle utilizando Anylogic.
Importancia	Vital.
Estado	Terminado.
Estabilidad	Alta.

Cuadro 1.1: Objetivos del Sistema: OBJ-001

Código	SUBOBJ-002
Descripción	Generar fichero con el código fuente del modelo de simulación.
Importancia	Vital.
Estado	Terminado.
Estabilidad	Alta.

Cuadro 1.2: Objetivos del Sistema: SUBOBJ-002

Código	SUBOBJ-003
Descripción	Aplicar estrategia de gamificación a un curso de Moodle.
Importancia	Importante.
Estado	Terminado.
Estabilidad	Alta.

Cuadro 1.3: Objetivos del Sistema: SUBOBJ-003

Código	SUBOBJ-004
Descripción	Dar persistencia a resultados de aplicar la estrategia de gamificación en el sistema externo SmartWeb.
Importancia	Importante.
Estado	Terminado.
Estabilidad	Alta.

Cuadro 1.4: Objetivos del Sistema: SUBOBJ-004

1.3. Acrónimos y Definiciones

A continuación, se proporcionan una serie de acrónimos y definiciones, que serán utilizadas durante el proyecto, para facilitar la comprensión del mismo.

- **Java:** Es un lenguaje de programación de propósito general, concurrente, orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo.
- **JavaFx:** Es una familia de productos y tecnologías de Sun Microsystems, adquirida por Oracle Corporation, para la creación de aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas.

  Las aplicaciones JavaFX pueden ser ejecutadas en una amplia variedad de dispositivos. JavaFX un lenguaje declarativo, además puede integrarse código Java en programas JavaFX. JavaFX es compilado a código Java, por lo que las aplicaciones JavaFX pueden ser ejecutadas en computadores con la máquina virtual de Java instalada.
- **SVN:** Apache Subversion, abreviado frecuentemente como SVN, es una herramienta de control de versiones open source basada en un repositorio cuyo funcionamiento se asemeja enormemente al de un sistema de ficheros. Es software libre bajo una licencia de tipo Apache/BSD.
- **Repositorio:** Es un sitio centralizado donde se almacena y mantiene información digital, habitualmente bases de datos o archivos informáticos.
- **Anylogic:** Es una herramienta desarrollada por The AnyLogic Company que incluye todos los métodos de simulación más comunes en practica hoy.
- **MBA:** Un modelo basado en agentes (MBA) es un tipo de modelo compu-

tacional que permite la simulación de acciones e interacciones de individuos dentro de un entorno.

- **Gamificación:** Es el uso de técnicas, elementos y dinámicas propias de los juegos y el ocio en actividades no recreativas con el fin de potenciar la motivación, así como de reforzar la conducta para solucionar un problema u obtener un objetivo.
- **Lenguaje XML (Extensible Markup Language):** Permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.
- **Eclipse:** Es un programa informático compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar aplicaciones. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados (del inglés IDE), como el IDE de Java llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse y que son usados también para desarrollar el mismo Eclipse. Sin embargo, también se puede usar para otros tipos de aplicaciones cliente, como BitTorrent o Azureus.
- **Metodología de desarrollo:** Metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.
- **Kanban:** Es un sistema de información que controla de modo armónico la fabricación de los productos necesarios en la cantidad y tiempo necesarios en cada uno de los procesos que tienen lugar tanto en el interior de la fábrica, como entre distintas empresas.

  También se denomina "sistema de tarjetas", pues en su implementación más sencilla utiliza tarjetas que se pegan en los contenedores de materiales y que se despegan cuando estos contenedores son utilizados, para asegurar la reposición de dichos materiales. Las tarjetas actúan de testigo del proceso de producción. Otras implementaciones más sofisticadas utilizan la misma filosofía, sustituyendo las tarjetas por otros métodos de visualización del flujo.
- **IDE:** Integrated Development Environment (IDE),entorno de desarrollo integrado, es una aplicación informática que proporciona servicios integrales para facilitarle al programador el desarrollo de software.

  Normalmente, un IDE consiste de un editor de código fuente, herramientas de construcción automáticas y un depurador. La mayoría de los IDE tienen auto-completado inteligente de código.
- **SGBD:** Sistema de gestión de base de datos, es una agrupación de programas que sirven para definir, construir y manipular una base de datos.
  - Definir una base de datos: consiste en especificar los tipos de datos, estructuras y restricciones para los datos que se almacenarán.

- Construir una base de datos: es el proceso de almacenar los datos sobre algún medio de almacenamiento.
- Manipular una base de datos: incluye funciones sobre la base de datos como pueden ser, una consulta, actualización, etc.

Capítulo 2

Generalidades

2.1. Introducción

En este capítulo se describen conceptos necesarios para entender mejor el ámbito de este proyecto. Este trabajo tiene como pilares fundamentales la gamificación y la simulación, que son conceptos innovadores, y por ello, la necesidad de explicarlos.

2.2. Introducción a la simulación

La simulación es la replicación de un sistema dado mediante el uso de métodos artificiales. Puede ser puramente analítico (a través de ecuaciones matemáticas) o computacional. Aunque es posible replicar las actividades de baja y mediana complejidad analítica, es extremadamente difícil construir modelos flexibles y obtener resultados precisos, por lo tanto, el uso de ordenadores se convierte en esencial.

Las simulaciones se pueden utilizar para comprender mejor el impacto de decisiones específicas, mediante el uso de la simulación por ordenador de sistemas reales. Las simulaciones por ordenador se pueden utilizar también en los centros educativos con el fin de desarrollar habilidades específicas, en las que los estudiantes controlan parte de las variables de simulación por ordenador a través de interfaces de usuario.

2.2.1. Modelado

El modelado es una manera con la que podemos resolver problemas del mundo real. En muchos casos, no podemos permitirnos experimentar con objetos reales para encontrar las soluciones adecuadas: construcción, destrucción, y haciendo cambios pueden ser demasiado caro, peligroso, o simplemente imposible. Si ese es el caso, podemos construir un modelo que utiliza un lenguaje de modelado para representar

el sistema real. Este proceso supone la abstracción: incluimos los detalles que creemos que son importantes y dejar de lado los que creemos que no son importantes. El modelo es siempre menos complejo que el sistema original.

Los modelos de simulación requiere de herramientas especiales de software que utilizan lenguajes específicos para la simulación. Es muy complejo realizar bien un modelo de simulación, pero su tiempo y esfuerzo serán recompensados cuando su modelo ofrezca un análisis de alta calidad de un sistema dinámico.

2.2.2. Métodos en los modelos de simulación

En los modelos de simulación, un método es un marco que usamos para trazar un sistema del mundo real para su modelo. Se puede pensar en un método como un tipo de lenguaje o una especie de términos y condiciones para la construcción de modelos.

Hay tres métodos:

- Dinámica de sistemas
- Modelado de eventos discretos
- Modelado basado en agentes

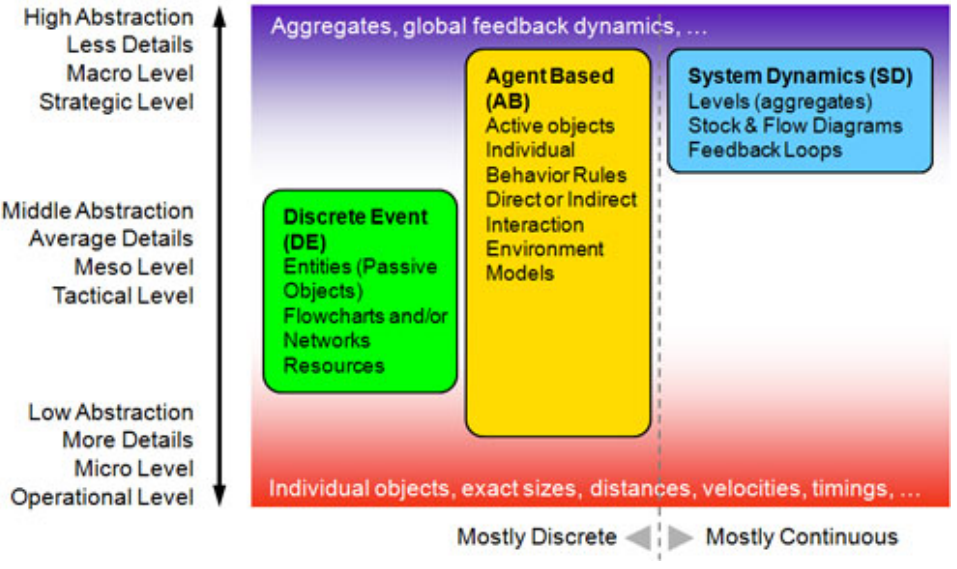

Figura 2.1: Enfoques de modelado en la escala de abstracción

Según se observa en la Figura 2.1, cada método de modelado tiene un rango específico de niveles de abstracción. La dinámica de sistemas asume la abstracción muy alta, y que por lo general se utiliza para el modelado estratégico. El modelado de eventos discretos da soporte a la abstracción media y media-baja. En el medio están los modelos basados en agentes, que pueden variar a partir de modelos muy detallados

donde los agentes representan objetos físicos para los modelos altamente abstractos donde los agentes representan a empresas de la competencia o de los gobiernos.

Siempre se debe seleccionar el método después de que se hayan examinado cuidadosamente el sistema que se desea modelar y sus objetivos.

Modelo basado en agentes (MBA)

Dado que para la naturaleza de esta aplicación se va a utilizar el enfoque MBA, en este apartado se describe en profundidad este tipo de modelado.

Un modelo basado en agentes (MBA) es un tipo de modelo computacional que permite la simulación de acciones e interacciones de individuos autónomos dentro de un entorno, y permite determinar qué efectos producen en el conjunto del sistema. Combina elementos de teoría de juegos, sistemas complejos, emergencia, sociología computacional, sistemas multi-agente, y programación evolutiva. Los modelos simulan las operaciones simultáneas de entidades múltiples (agentes) en un intento de recrear y predecir las acciones de fenómenos complejos. Es un proceso de emergencia desde el nivel más elemental al más elevado.

El modelado basado en agentes es un método relativamente nuevo en comparación con la dinámica de sistemas y el modelado de eventos discretos. De hecho, el modelado basado en agentes fue en gran medida un tema académico hasta que los practicantes de simulación lo comenzaron a usar hace unos 15 años.

Se desencadena por:

- Una deseo de obtener una visión más profunda en los sistemas que los métodos de modelado tradicionales no captan bien.
- Los avances en la tecnología de modelado posibles gracias a la informática, tales como el modelado orientado a objetos, UML, y gráficos de estado.
- El rápido crecimiento de la potencia de la CPU y la memoria. Los modelos basados en agentes son más exigentes que la dinámica de sistemas y modelos de eventos discretos.

No hay lenguaje estándar para el modelado basado en agentes, y la estructura de un modelo basado en agentes proviene de editores gráficos o secuencias de comandos. Hay muchas formas de especificar el comportamiento de un agente. Con frecuencia agente tiene una noción de estado y sus acciones y reacciones dependen del estado, entonces el comportamiento se define mejor con gráficos de estado. A veces, el comportamiento se define en las reglas ejecutadas en eventos especiales.

En muchos casos, la mejor manera de capturar la dinámica interna del agente es el uso de la dinámica de sistemas o un enfoque de eventos discretos, y después colocar un diagrama de flujo dentro de un agente. Del mismo modo, los agentes externos a la dinámica del entorno en el que se encuentran a menudo se modelan de forma normal utilizando métodos tradicionales. Es por eso que muchos modelos basados en agentes son modelos multi-método.

Los agentes en un modelo basado en agentes pueden representar cosas muy diversas: vehículos, unidades de equipos, proyectos, productos, ideas, organizaciones, inversiones, parcelas de tierra, las personas en diferentes roles, etc.

2.2.3. Herramientas software de simulación

Actualmente podemos encontrar una gran variedad de programas de simulación para la mayoría de industrias, manejando proyectos complejos, y a gran escala, relacionados con la fabricación, procesos, logística, distribución, almacenamiento o sistemas de servicios. Entre los más conocidos podemos encontrar:

- **Promodel** es una tecnología de simulación de eventos discretos que se utiliza para planificar, diseñar y mejorar nuevas o existentes de fabricación, logística y otros sistemas operativos. Representa con precisión los procesos del mundo real, incluyendo su variabilidad inherente y las interdependencias, con el fin de realizar un análisis predictivo de los cambios potenciales.
- **Arena** ha sido durante 30 años el software de simulación de eventos discretos líder en el mundo. La simulación de eventos discretos describe un proceso con un conjunto de eventos únicos, específicos en el tiempo. Estos modelos flexibles, basados en la actividad pueden ser utilizados eficazmente para simular casi cualquier proceso.
- **Anylogic** es considerada como una herramienta de simulación de última generación. Un software que ofrece un sistema flexible, orientado a objetos para el desarrollo de la simulación de eventos discretos, modelos basados en agentes y de dinámica de sistemas. Los modelos de AnyLogic se pueden exportar como applets Java o aplicaciones Java autónomas (versión Professional) sin necesidad de un entorno de desarrollo AnyLogic en funcionamiento.

2.2.4. Anylogic

De los diferentes software de simulación presentados en el apartado anterior el elegido para realizar este proyecto es el software de simulación Anylogic. Los motivos de esta elección han sido básicamente dos:

- A pesar de ser un software muy completo, su aprendizaje es muy sencillo. Tiene una interfaz muy atractiva para el usuario y su compatibilidad con el lenguaje Java es total.
- Permite, con una cierta facilidad, generar desde otras aplicaciones, modelos ejecutables. Estos modelos son construidos a través de ficheros XML que contienen el código tanto del modelo como de la interfaz.
- Contar con el apoyo de la tutora de este proyecto, que ya tenía conocimientos previos de la herramienta, así como la posibilidad de contar con otros alumnos que han trabajado con Anylogic en ocasiones anteriores.

La versión utilizada para realizar todos los experimentos va a ser la PLE.

2.2.5. Ventajas

A continuación las principales ventajas de practicar la simulación.

- Un modelo de simulación permite analizar sistemas y encontrar soluciones donde los métodos tales como cálculos analíticos y de programación lineal fallan.
- Una vez que haya elegido un nivel de abstracción, es más fácil de desarrollar un modelo de simulación que un modelo analítico.
- La estructura de un modelo de simulación refleja, naturalmente, la estructura del sistema.
- En un modelo de simulación, se puede medir los valores y se pueden añadir mediciones y análisis estadísticos en cualquier momento.
- Los modelos de simulación son mucho más convincentes que las hojas de cálculo.

2.3. Introducción a la gamificación

2.3.1. Definición

Se puede definir el concepto de gamificación como el uso de técnicas, elementos y dinámicas propias de los juegos y el ocio en actividades no recreativas con el fin de potenciar la motivación, así como de reforzar la conducta para solucionar un problema u obtener un objetivo.

2.3.2. Ámbitos de aplicación

Actualmente, la gamificación tiene muchos ámbitos de aplicación, entre los que se encuentran:

- Salud, existen juegos que ayudan a ejercitar la memoria, la agilidad mental, etc.
- Empresas, cada vez hay mas empresas que aplican la gamificación para promover la participación de los empleados en actividades corporativas, para mantener la motivación, promover el trabajo colaborativo, aumentar la productividad, etc.

Aunque, si hay que destacar un ámbito de aplicación de la gamificación, ese es el de la educación. El aprendizaje es uno de los procesos que podrían considerarse como más

complicados y aburridos, al menos desde el punto de vista del alumno, considerando los modelos de enseñanza tradicional. Por ello es importante considerar la gamificación como una alternativa de apoyo en los procesos de enseñanza-aprendizaje debido a las siguientes razones:

- Estimula y hace más atractiva la participación de los estudiantes.
- Simplifica las actividades difíciles.
- Motiva la participación constante.
- Crea una retroalimentación positiva a través de recompensas.
- Promueve la perseverancia y el triunfo.
- Aumenta el compañerismo.
- Transforma actividades aburridas en divertidas e interesantes.
- Fomenta la comunicación entre pares.
- Crea ambientes de confianza.

A pesar de los beneficios, nunca debemos olvidar que la gamificación es solo una estrategia de apoyo en el aula y que las dinámicas de juego nunca deben sustituir el verdadero propósito de los procesos de aprendizaje. El contenido académico debe ser el principal motor que impulse las mecánicas del juego, de ahí la responsabilidad del docente para controlar y mediar la gamificación en el área educativa.

2.3.3. Estrategias de gamificación

En este apartado se describirán los principales elementos de una estrategia de gamificación.

Mecánicas de gamificación

Como cualquier juego que se preste, la gamificación no está exenta de unas normas de funcionamiento. Dichas mecánicas permiten que los usuarios adquieran un compromiso para superar los distintos retos a los que se someten. De entre las mecánicas que más aceptación tienen se pueden destacar:

- Colección: Se parte de la importancia que tiene para los usuarios los logros y las recompensas.
- Puntos: Muy usados para conseguir la fidelización de los usuarios en la tarea que se les ha sido asignada.
- Ranking: Se establece una clasificación o comparación entre los usuarios.
- Nivel: Muy comunes en los deportes como, por ejemplo, el fútbol o el baloncesto. Los niveles dan fe de los progresos de los usuarios en las actividades a

las que han sido asignadas.

- Progresión: La progresión es otra técnica muy usual en la gamificación y consiste en completar el 100 % de la actividad que se ha encomendado. Publicar estos progresos en redes sociales es una práctica habitual.

Dinámicas de gamificación

Las dinámicas de juego son un aspecto indispensable para la elaboración de cualquier actividad relacionada con la gamificación.

Si antes se explicaba la importancia de que los usuarios conozcan las normas de cualquier juego, en este caso de cara a la gamificación se hace imprescindible que los usuarios tengan perfectamente asimiladas qué dinámicas de juego se llevarán a cabo. Dichas dinámicas de juego tienen por objeto la motivación y la implicación en la realización de una actividad.

A través de las dinámicas de juego, se consigue despertar el interés de los usuarios por las actividades que están llevando a cabo. Entre las dinámicas más conocidas destacan:

- Recompensa: La recompensa en una actividad no tiene otra función que despertar el interés por el juego.
- Competición: Aunque no siempre es vista como una cualidad positiva, la buena gestión de la competición es un magnífico instrumento para atraer el interés del usuario por una actividad. Además, dicha competición tiene la ventaja de poder realizarse de forma individual, por parejas o en grupo.
- Estatus: El estatus logrado a través de la gamificación incentiva enormemente al usuario en la consecución y realización de la actividad que se le ha encomendado.
- Cooperación: Se trata de otra forma de competir, pero en este caso se juega con el hecho de que es un mismo grupo el que persigue un mismo fin.
- Solidaridad: Se trata de una dinámica muy interesante y muy ligada con la cooperación. Mediante la solidaridad se fomenta la ayuda mutua entre compañeros y de una manera altruista, es decir, sin esperar ninguna recompensa a cambio.

Proceso de gamificación

Para realizar un proceso de gamificación exitoso, es imprescindible seguir ciertas pautas.

- Viabilidad: En primer lugar hay que valorar si la gamificación es aplicable. Hay ciertos campos en los que la aplicación de la gamificación no es exitosa.

- Objetivos: Hay que definir cuáles serán los objetivos de la gamificación. Es muy importante tener claro los objetivos que se persiguen con la aplicación de esta técnica.
- Motivación: Otro aspecto a valorar es la predisposición y el perfil de las personas con las que se va a llevar a cabo la gamificación en una actividad.
- Resultados: Es imprescindible realizar una evaluación de los resultados de la propuesta de gamificación que se haya llevado a cabo.

Capítulo 3

Planificación del Proyecto

En esta sección del documento se describen todos los aspectos relativos a la gestión del proyecto: metodología, fases del proyecto, costes, riesgos y planificación.

3.1. Metodología

Este proyecto se ha desarrollado siguiendo una metodología iterativa e incremental [Véase en Figura 3.1]. Cada iteración ha tenido su fase de análisis, diseño, desarrollo y pruebas. Se ha elegido esta metodología de desarrollo porque era la mejor manera de agilizar el desarrollo de la aplicación, teniendo así versiones funcionales y permitiendo un mayor feedback con la tutora del proyecto. Esto ha permitido hacer pequeños cambios después de cada iteración que no han repercutido mucho en el tiempo de desarrollo.

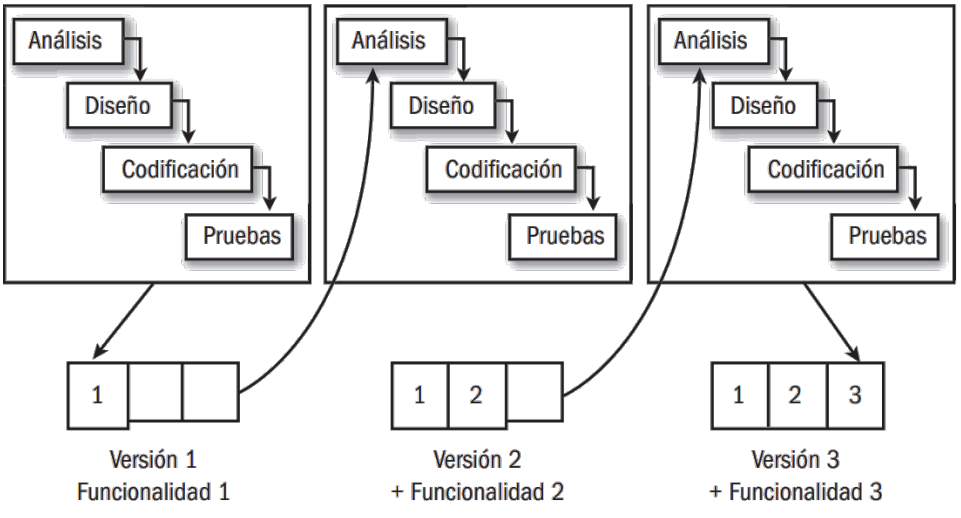

Figura 3.1: Metodología Iterativo e Incremental

Para la organización y desglose de las tareas de cada iteración se ha utilizado el sistema Kanban, teniendo así un gran control de las tareas que están pendientes, las que se están realizando y de las que están finalizadas. Para este sistema, se ha utilizado la herramienta online Trello[13].

3.2. Fases del Proyecto

Se ha realizado una fase previa de gran importancia, realizando el aprendizaje necesario para familiarizarse con las técnicas de gamificación, para la construcción de un modelo de simulación y para el uso de la herramienta Anylogic. Posteriormente, se han realizado tres iteraciones con versiones funcionales de la aplicación. Cada iteración incrementa y mejora la funcionalidad de la iteración anterior.

A continuación, se describirán cada una de las fases por la que ha pasado el proyecto:

3.2.1. Adquisición de conocimientos

La adquisición de conocimientos necesarios para el desarrollo del proyecto se ha llevado a cabo en todas las fases del proyecto desde el inicio hasta el final.

En la documentación de cada fase del proyecto, se especifica con detalle el aprendizaje adquirido durante cada etapa.

3.2.2. Iteración 0: Prototipo de modelo de simulación

Durante la fase previa del proyecto, se construyó un modelo de simulación base. Para ello, se realizaron las siguientes tareas:

- Aprendizaje de la herramienta de simulación Anylogic, y de los conceptos necesarios para entender la simulación basada en agentes.
- Toma de decisión de las variables, parámetros, fórmulas, gráficas, etc. que se han utilizado para la construcción del modelo.
- Construcción de un modelo de simulación en Anylogic que posteriormente se utilizaría como base para la construcción dinámica de los modelos de simulación.
- Realizar una API de los distintos elementos de Anylogic que se han usado en la simulación, con su correspondiente código XML para su generación dinámica.
- Validación del modelo base, comprobando que tiene un comportamiento razonable y acorde a los parámetros de entrada que recibe.

3.2.3. Iteración 1: Lectura de estrategias

Durante la iteración 1 del proyecto, se desarrolló el módulo para la lectura de las estrategias guardadas en un archivo XML. Para ello, se realizaron las siguientes tareas:

- Aprendizaje de JavaFx para empezar a desarrollar la aplicación, aprendizaje de la librería JDOM para la lectura y escritura de un archivo XML desde java.
- Toma de decisión de la estructura del archivo XML en el cual se encuentran las estrategias de gamificación.
- Desarrollo de los objetos y clases necesarios para la utilización de la información obtenida en el archivo XML.
- Desarrollo de una interfaz de usuario base para probar la funcionalidad de leer el archivo XML, mostrando en pantalla las estrategias leídas y la interacción con éstas.
- Pruebas de lectura del archivo XML.

3.2.4. Iteración 2: Generar del modelo de simulación

Durante la iteración 2 del proyecto se desarrolló el módulo para generar el archivo de simulación. Para ello, se realizaron las siguientes tareas:

- Desarrollo de interfaz de usuario para la generación del modelo de simulación. Se ha desarrollado una pantalla con un formulario para que el usuario introduzca los parámetros de entrada necesarios para la simulación.
- Desarrollo de los algoritmos necesarios para construir el modelo de simulación dinámicamente.
- Validación de los campos del formulario, controlando que se reciben los parámetros en el formato deseado, y en caso contrario mostrando ventanas informando al usuario.
- Pruebas de la creación del archivo con extensión .ALP.
- Verificación que el archivo generado se abre sin problemas en Anylogic, y que el modelo cumple con lo esperado.
- Pruebas de la ejecución correcta del modelo en Anylogic.

3.2.5. Iteración 3: Aplicar estrategia a Moodle

Durante la iteración 3 del proyecto se desarrolló el módulo para aplicar la estrategia a Moodle y posteriormente guardar los resultados obtenidos. Para ello, se realizaron las siguientes tareas:

- Instalación y aprendizaje del entorno de Moodle.
- Conexión a Moodle utilizando JDBC.
- Aprendizaje de librerías Jersey para la utilización de servicios web REST.
- Desarrollo de los algoritmos necesarios para construir las consultas a base de datos para recuperar la información necesaria y su posterior utilización.
- Desarrollo de la interfaz de usuario para esta funcionalidad.
- Pruebas de conexión con Moodle, así como la gestión de los posibles errores.
- Pruebas de los servicios REST, así como la gestión de los posibles errores.

3.3. Riesgos

En esta sección se realizará un análisis de los riesgos contemplados en el desarrollo de esta aplicación. A continuación, en los Cuadros 3.1, 3.2, 3.3, 3.4 y 3.5 se describirán cada uno de los riesgos contemplados durante el desarrollo de este proyecto y en el posterior uso de la aplicación:

ID Riesgo: R-001	Impacto: Muy alto
Probabilidad	5 %
Descripción	Actualizaciones de Anylogic.
Contexto	Anylogic suele cambiar la estructura de la cabecera de sus archivos, al hacer actualizaciones de software. Esto podría suponer incompatibilidades con los archivos generados por la aplicación.
Reducción	Si se produce alguna actualización durante el desarrollo, hay que comprobar si ha habido alguna modificación que perjudique a la aplicación en desarrollo.
Plan de contingencia	Adaptarse a posibles cambios con la menor repercusión posible.
Estado actual	Este riesgo será constante durante todo el tiempo que esté la aplicación en uso.

Cuadro 3.1: Riesgos: R-001

ID Riesgo: R-002	Impacto: Medio-Alto
Descripción	Inestabilidad de Anylogic en sistemas MAC OS.
Probabilidad	40 %
Contexto	Anylogic es bastante inestable en MAC OS.
Reducción	No es posible reducir la probabilidad de que ocurra.
Plan de contingencia	Tener una versión de Anylogic en Windows.
Estado	Este riesgo solo estará presente para usuarios de la aplicación en sistemas MAC OS.

Cuadro 3.2: Riesgos: R-002

ID Riesgo: R-003	Impacto: Medio
Probabilidad	10 %
Descripción	Escritura de los resultados de aplicar una estrategia a Moodle en base de datos.
Contexto	La base de datos se encuentra en un servidor al cual se accede con peticiones REST, es posible que el servidor falle en algún momento.
Reducción	Esta parte es externa a este proyecto, luego no se puede reducir la probabilidad de que ocurra.
Plan de contingencia	Controlar el fallo e informar al usuario y al administrador del sistema externo.
Estado	Este riesgo será constante durante todo el tiempo que esté la aplicación en uso.

Cuadro 3.3: Riesgos: R-003

ID Riesgo: R-004	Impacto: Medio
Probabilidad	5 %
Descripción	Ninguna experiencia con simulación.
Contexto	Nunca antes se había tenido contacto con la simulación.
Reducción	Tener material disponible para cualquier consulta.
Plan de contingencia	Realizar un buen aprendizaje.
Estado	No está contemplado.

Cuadro 3.4: Riesgos: R-004

ID Riesgo: R-005	Impacto: Alto
Probabilidad	10 %
Descripción	No explotación de la aplicación.
Contexto	Si los resultados de la simulación no se acercan a la realidad es posible que la aplicación no llegue a ser usada.
Reducción	Hacer un buen estudio del comportamiento de los agentes.
Plan de contingencia	Hacer numerosos experimentos para observar los resultados e intentar acercarse lo máximo posible a la realidad.
Estado	Este riesgo estará presente durante la explotación de la aplicación.

Cuadro 3.5: Riesgos: R-005

3.4. Distribución Temporal

A continuación, se muestra la distribución temporal del proyecto. Para ello, se muestran dos diagramas de Gantt, en el primero se muestra la primera estimación temporal realizada [Véase en Figura 3.2], y en el segundo la duración real del proyecto [Véase en Figura 3.3].

Según la primera estimación el proyecto duraría en total 10 meses aproximadamente. Esta primera estimación se hizo teniendo en cuenta la finalización de todas las asignaturas del Grado de Ingeniería Informática y estimando una dedicación plena a este proyecto.

Por motivos laborales, la dedicación al proyecto no ha sido la que se estimó en primera instancia y, finalmente, la duración total del proyecto ha sido de 19 meses, aproximadamente. Cabe decir que durante este tiempo no ha habido una dedicación regular, habiendo periodos de más carga de trabajo y otros periodos de menos. En el segundo diagrama, es donde podemos ver representada la duración real de cada una de las tareas realizadas para el desarrollo de este proyecto.

Ambos diagramas se han estructurado de la misma manera, en ellos se pueden observar 4 iteraciones, descritas en apartados anteriores, compuestas por la fase de análisis, la fase de diseño, la fase de implementación y la fase de pruebas. Estas tareas, en algunos casos, se solapan en el tiempo dado que no dependían unas de otras. También hay una tarea de aprendizaje que prácticamente abarca toda la duración del proyecto y, por último, el tiempo dedicado a la documentación.

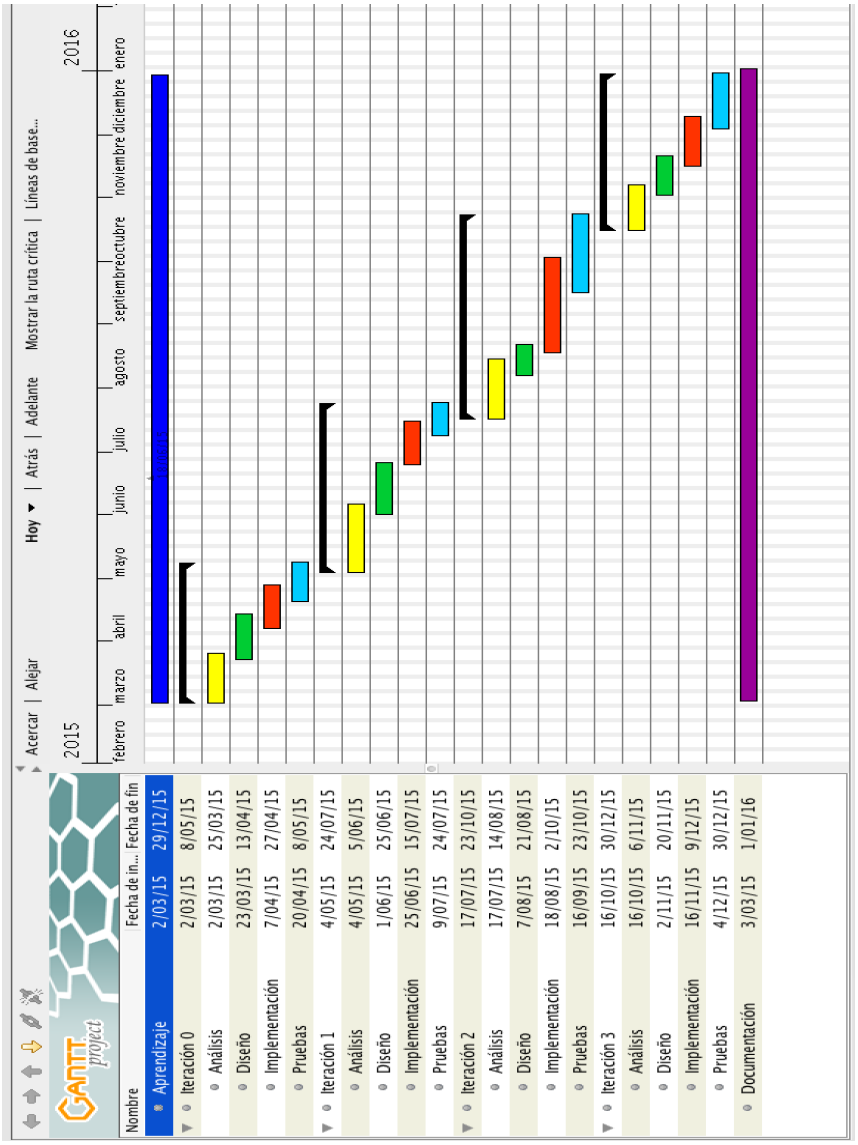

Figura 3.2: Distribución Temporal Estimada

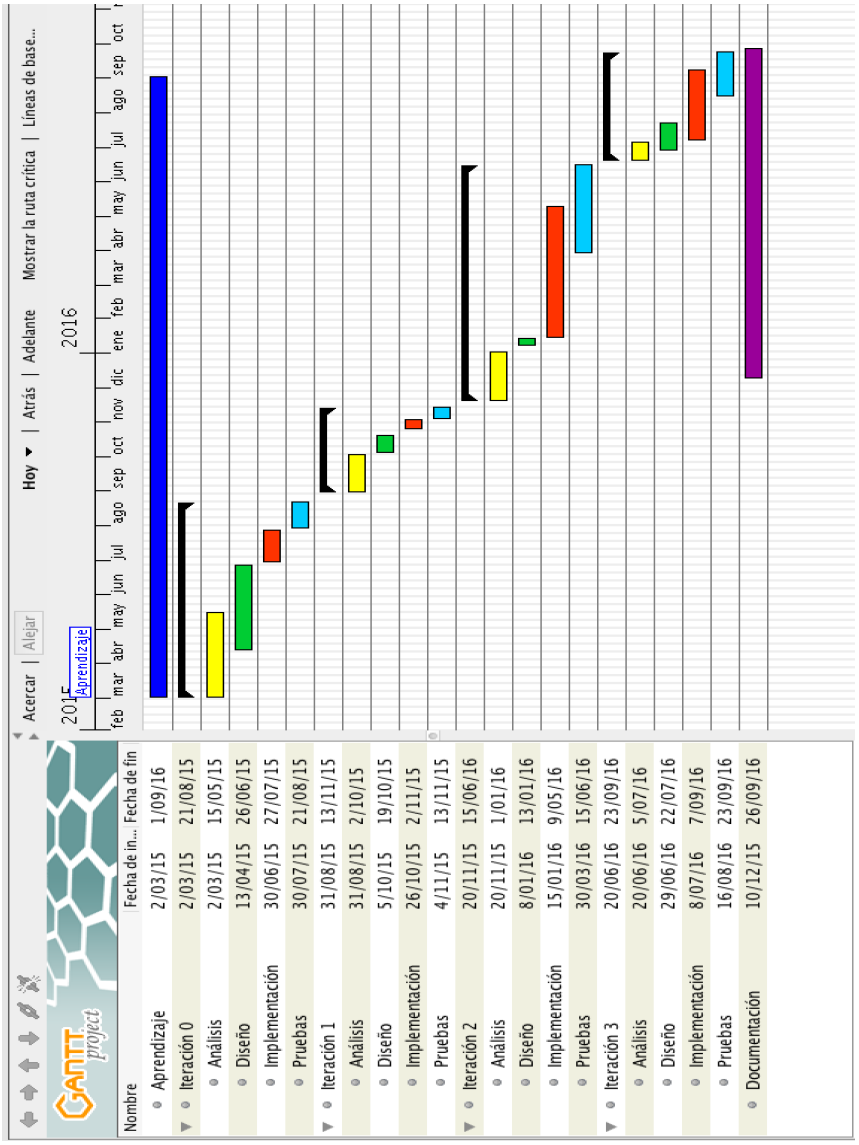

Figura 3.3: Distribución Temporal Real

3.5. Costes

En este apartado se analizará los costes necesarios para el desarrollo de este proyecto. El coste total del proyecto se ha calculado con la siguiente fórmula:

$$CosteTotal = CostesDirectos + CostesIndirectos$$

3.5.1. Costes Directos

Según el Diagrama de Gantt expuesto en la Figura 3.3, el proyecto se ha realizado en 19 meses. Quitando fines de semana tenemos un total de 380 días, estableciendo una media de trabajo de 2 horas/día. En total serían 760 horas de trabajo.

Estableciendo una categoría de Analista-Programador, según las tablas salariales del convenio para las empresas de Tecnologías de la Información y la Comunicación, publicado en el Boletín oficial del Estado de abril de 2009 y vigentes a septiembre 2016[22], el sueldo medio es de 10 euros/hora. Por lo que tenemos un coste:

$$CostesDirectos = 760horas \times 10euros/hora$$

$$CostesDirectos = \mathbf{7600\ €}$$

3.5.2. Costes Indirectos

En este proyecto no ha sido necesario ningún coste en material. En lo referente al hardware, sólo ha sido necesario un portátil, propiedad del desarrollador del proyecto. Las licencias de casi todo el software utilizado son libres, en el caso de algún software sin licencia libre se ha utilizado versiones de prueba, por lo tanto no ha requerido coste alguno.

Respecto a gastos de electricidad, según los cálculos realizados anteriormente se ha dedicado a este proyecto 760 horas de trabajo, que equivale aproximadamente a 31 días. Con una media del gasto obtenido en las facturas de electricidad se ha realizado un cálculo de cuanto habría supuesto estos 31 días de trabajo. Este gasto asciende a 100 €.

$$CostesIndirectos = \mathbf{100\ €}$$

3.5.3. Coste Total

En total tenemos que el coste total del proyecto es:

$$CosteTotal = \textbf{7600 €} + \textbf{100 €}$$

$$CosteTotal = \textbf{7700 €}$$

3.6. Precio

El precio final de la aplicación sera la suma de los costes totales, el margen de beneficio y el 21 % de IVA que fija la ley en España. Se ha fijado un margen de beneficio de 1500 €.

$$Precio = \textbf{7700 €} + \textbf{1500 €} + IVA$$

$$Precio = \textbf{11132 €}$$

Capítulo 4

Estudio del sistema

Para construir el modelo de un sistema real, es necesario recorrer una serie de etapas de modelado. Se comienza haciendo una descripción del sistema real, para ello es necesario realizar un estudio de toda la información disponible. Entre otras cosas, es necesario identificar cuáles serán las entradas y las salidas de nuestro modelo. Una vez identificadas las diferentes partes del sistema real se construye un modelo conceptual que contenga todos los elementos que se consideren relevantes del sistema. Desde este se pasa a un modelo lógico que contiene las relaciones lógicas entre los elementos. Por último se construye el modelo de ordenador (o modelo de simulación) que ejecuta la lógica recogida en la fase anterior.

4.1. Descripción del sistema

Actualmente en la Universidad de Cádiz, se utiliza Moodle como plataforma de aprendizaje, un sistema de gestión de cursos, que ayuda a los educadores a crear comunidades de aprendizaje en línea. Este tipo de plataformas tecnológicas también se conocen como LCMS (Learning Content Management System).

Estas herramientas son de gran utilidad en el ámbito educativo, ya que permiten a los profesores la gestión de cursos virtuales para sus alumnos, o la utilización de un espacio en línea que dé apoyo.

A pesar de ser una herramienta muy potente, tiene una principal desventaja. Algunas actividades pueden ser un poco mecánicas, por lo que puede crear desmotivación entre los alumnos y provocar que el uso de la plataforma no sea el esperado. Con el fin de buscar una solución a este problema, realizamos este caso de estudio, en el que se estudiará la posibilidad del uso de técnicas, elementos y dinámicas propias de los juegos en la plataforma Moodle con el fin de potenciar la motivación de los alumnos para su uso.

4.2. Formulación del problema

En este apartado se va a definir de forma detallada los objetivos, las restricciones y las variables que se van a utilizar para definir el modelo de simulación.

El modelo a diseñar debe de ser capaz de simular el proceso por el cual un alumno va realizando una secuencia de actividades, diseñadas por el profesor. Para ello el profesor introduce una serie de parámetros de entrada al modelo, estos parámetros determinarán el comportamiento de los alumnos en la simulación.

Para la modelado del sistema, se utilizará el modelado basado en agentes. Este tipo de modelos se componen principalmente de una población de agentes y de una serie de estados por el cual pasarán los agentes. A continuación se describirá los elementos que componen este sistema:

Agente

El modelo tendrá una población de agentes que representará a los alumnos participantes del curso. Cada agente representa a un alumno, que tendrá un comportamiento propio dentro del sistema dependiendo de una serie de parámetros. Se ha decidido que cada agente tenga una variable que represente la motivación de este y que dependiendo de esta el alumno tenga un comportamiento u otro dentro del sistema.

Estados

Los agentes de este modelo pasarán por una serie de estados [Véase en Figura 4.1]. Estos estados representan la secuencia de actividades diseñadas por el profesor para la estrategia de gamificación. Cuando un agente se encuentra en un determinado estado, se entenderá que se encuentra realizando la actividad que este estado representa.

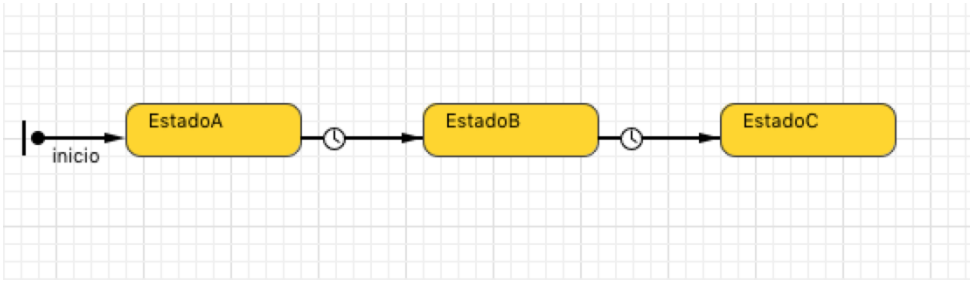

Figura 4.1: Ejemplo Estados

Cuando el agente se encuentra en el Estado C, significa que esta realizando la actividad que este estado representa, y que ha superado con éxito las actividades que representan el Estado A y el Estado B.

Como uno de los principales objetivos es aplicar el concepto de gamificación, es muy importante que tras la finalización con éxito de una actividad el alumno reciba una recompensa, para así premiar el esfuerzo realizado y reforzar su motivación de cara

a la superación de próximas tareas. Es por eso, que cada actividad llevará asignada una serie de recompensas o premios, que se le concederán a aquellos alumnos que la superen con éxito.

Nuestro modelo contará con los siguientes tipos de recompensas :

- **Puntos**, representa la recompensa con menor valor y se puede asignar cuando el alumnos realiza una actividad que no requiere de gran esfuerzo.
- **Puntos de experiencia**, estos se sólo se obtienen cuando un usuario ha completado una subsecuencia de la estrategia.
- **Puntos de Level-up**, al igual, que antes, sólo se pueden asignar cuando se ha completado una subsecuencia de la estrategia, y significaría subir de nivel.
- **Badges**, representa una medalla. Este tipo de recompensa tiene un alto valor y puede ser utilizado en actividades que tengan un gran peso en el transcurso de la estrategia: exámenes, trabajos finales, practicas, etc.
- **Random**, si el diseñador de la estrategia elige esta opción, indica que aleatoriamente se asignará una recompensa que él puede elegir. Esta recompensa debería de ser de bajo nivel como por ejemplo puntos.
- **Puntos First-time**, el diseñador podrá elegir esta recompensa si desea que tras realizar una determinada actividad por primera vez el usuario reciba una recompensa.

Transiciones

Las transiciones son la comunicación entre estados. En nuestro modelo tienen una gran importancia ya que son las encargadas de evaluar si un alumno ha realizado una actividad con éxito o no. Esta condición de guarda que evalúa si el alumno pasa al siguiente estado o no viene determinada por una probabilidad de éxito p, determinada por una tabla que describiremos más adelante. Donde la probabilidad de que el alumno no tenga éxito es $1 - p$.

En nuestro modelo la transición de un estado a otro es de tipo temporal, es decir, una vez que el agente supera la condición de guarda, lo único que le separa del siguiente estado es el tiempo. Este tiempo representa el empleado en realizar la actividad. En nuestro modelo la unidad de tiempo establecida es la hora, por lo que cualquier unidad de tiempo cuantifica el número de horas.

4.2.1. Objetivos

Los principales objetivos de este modelo de simulación son los siguientes:

- Estudiar el posible comportamiento de los alumnos ante una posible gamificación de un curso en la plataforma Moodle.
- Realizar un estudio de las actividades más propensas a ser abandonadas por

el alumno, es decir que no se completen con éxito.

- Obtener un balance general del tiempo medio empleado por los alumnos en las distintas actividades.
- Ver los logros conseguidos por los participantes del curso.

4.2.2. Variables

Para definir el modelo del sistema es necesario identificar las variables que puedan representar el funcionamiento del sistema real.

Parámetros de entrada

Nombre	Descripción
Número de alumnos	Número de alumnos que participarán en la simulación.
Motivación	Motivación que tendrán los alumnos.
Duración	Duración máxima de la simulación.
Distribución	La distribución que van a seguir los rangos de valores utilizados en la representación de la motivación y de la dificultad de las actividades.
Dificultad	El tipo de dificultad que se le asignará a cada actividad.
Tasa de éxito	Es la probabilidad de que un alumno realice las actividades con éxito.

Cuadro 4.1: Parámetros de entrada al modelo

Valores de los parámetros de entrada

A continuación se describirán los valores que pueden tener los parámetros que aparecen en el Cuadro 4.1.

- **Número de alumnos**, este parámetro será un número entero mayor que 0.
- **Distribución**, algunos de los parámetros de entrada son conceptuales, como por ejemplo la dificultad de las actividades o la motivación de los alumnos cuyo valor puede ser baja, media o alta. Detrás de estos conceptos hay unos valores numéricos para poder realizar el modelo. Para que el modelo tenga una cierta

flexibilidad asignamos a cada concepto un rango de valores que son asignados aleatoriamente, así por ejemplo todos los alumnos con motivación baja no tienen en su variable motivación el mismo valor. Este parámetro distribución establece cómo se van a distribuir los valores del rango. Para ello el usuario puede elegir entre tres opciones:

- Distribución uniforme : El rango se distribuye de manera equitativa [Véase en Cuadro 4.2].

Tipo	**Valor**
baja	1 a 10
media	11 a 20
alta	21 a 30

Cuadro 4.2: Distribución uniforme

- Distribución extremos mayores : El rango se distribuye de manera que rangos de los extremos (baja y alta) son mayores. Esto significa que puede haber alumnos con motivación baja o alta y que la diferencia del valor de su motivación sea notoria. Sin embargo, los alumnos con motivación media tendrán entre ellos valores más cercanos [Véase en Cuadro 4.3].

Tipo	**Valor**
baja	1 a 11
media	12 a 19
alta	20 a 30

Cuadro 4.3: Distribución extremos mayores

- Distribución extremos menores : El rango se distribuye de manera que rangos de los extremos (baja y alta) son menores. Es lo contrario al anterior. Esto significa que puede haber alumnos con motivación media y que la diferencia del valor de su motivación sea significativa. Sin embargo, los alumnos con motivación baja o alta tendrán entre ellos valores más cercanos [Véase en Cuadro 4.4].

Tipo	**Valor**
baja	1 a 7
media	8 a 23
alta	24 a 30

Cuadro 4.4: Distribución extremos menores

- **Motivación**, este parámetro esta subdividido en tres parámetros : motivación baja, motivación media y motivación alta. Cada uno de estos tres parámetros representan el porcentaje de alumnos que tienen una determinada motivación. El valor será un número entero, entre 0 y 100. La suma de los tres parámetros debe de ser 100 ya que todos los alumnos deben tener asignada una motivación. Finalmente el alumno tendrá una motivación cuyo valor dependerá de si es alta, baja o media y de la distribución elegida por el usuario.
- **Dificultad de las actividades**, cada actividad tendrá asignada una dificultad que podrá ser baja, media o alta. Esta variable tendrá un valor que dependerá de si es alta, baja o media y de la distribución elegida por el usuario.
- **Duración**, este parámetro puede tener 3 valores diferentes: duración mensual, duración cuatrimestral o duración anual. Correspondiente con el tiempo que el usuario desea simular.
- **Tasa de éxito**, esta variable es la que determina la capacidad de un alumno de realizar las actividades. Viene determinada por una tabla en un archivo Excel. Este archivo lo cargará el usuario en la aplicación. Se ha elegido un archivo Excel para que el usuario pueda configurar los valores de éxito para los alumnos, según sean sus intereses.

  El alumno, dependiendo de su motivación y de la dificultad de de la tarea que esté realizando, recibirá una probabilidad de éxito que será la que determinará si realiza la actividad con éxito o no.

  Como se puede observar en el Cuadro 4.5, se ha dado mayor peso a la motivación del alumno, ya que según la experiencia obtenida como estudiante pienso que un alumno motivado está más capacitado para realizar actividades con independencia de su dificultad.

  El usuario de la aplicación puede modificar la columna probabilidad de éxito para asignar los valores que crea oportunos.

Motivación	Dificultad de la actividad	Probabilidad de éxito
Alta	Baja	100 %
Alta	Media	90 %
Alta	Alta	80 %
Media	Baja	70 %
Media	Media	55 %
Media	Alta	40 %
Baja	Baja	30 %
Baja	Media	20 %
Baja	Alta	10 %

Cuadro 4.5: Tabla de tasa de éxito

4.2.3. Fórmulas

A continuación, mostramos las fórmulas utilizadas para el funcionamiento del modelo de simulación:

- Tiempo empleado en realizar una actividad

$\frac{DificultadActividad}{MotivacinAlumno} \times \alpha, \alpha \equiv$ número aleatorio del rango [0,1)

4.2.4. Datos de salida

El modelo de simulación genera los siguientes resultados:

Gráfica de abandonos

Esta gráfica, es un histograma, que contiene el número de alumnos que han abandonado la estrategia en cada actividad.

Gráfica de tiempos

Esta gráfica, es un histograma, que contiene el tiempo medio empleado por los alumnos en realizar cada actividad.

Gráfica de premios

Esta gráfico de sectores, contiene la cantidad de puntos, badges, puntos de first-time y puntos de experiencia que han conseguido los alumnos al terminar la estrategia.

Gráfica de niveles

Esta gráfico de sectores, contiene la cantidad de alumnos que han terminado la estrategia en cada nivel. Hay 10 niveles diferentes.

Por último también se indica el número de alumnos que han completado la estrategia con éxito.

4.3. Análisis y validación del modelo

Tras el modelado del problema, se ha realizado una batería de ejecuciones del modelo. Esto ha permitido realizar un análisis para determinar si el modelo generado es válido.

La validez del modelo se ha evaluado comprobando que éste se comporta de la manera esperada dependiendo de los parámetros introducidos por el usuario.

4.3.1. Análisis

A continuación, se describen las distintas simulaciones realizadas para la validación del modelo. Los datos descritos en las gráficas son datos promedios resultantes de realizar 10 simulaciones de cada tipo.

Escenario A

Datos:

- Población : 50 alumnos
- Alumnos con motivación baja : 10 %
- Alumnos con motivación media : 10 %
- Alumnos con motivación alta : 80 %
- Dificultad Actividad 1 : baja
- Dificultad Actividad 2 : media
- Dificultad Actividad 3 : alta

Resultados:

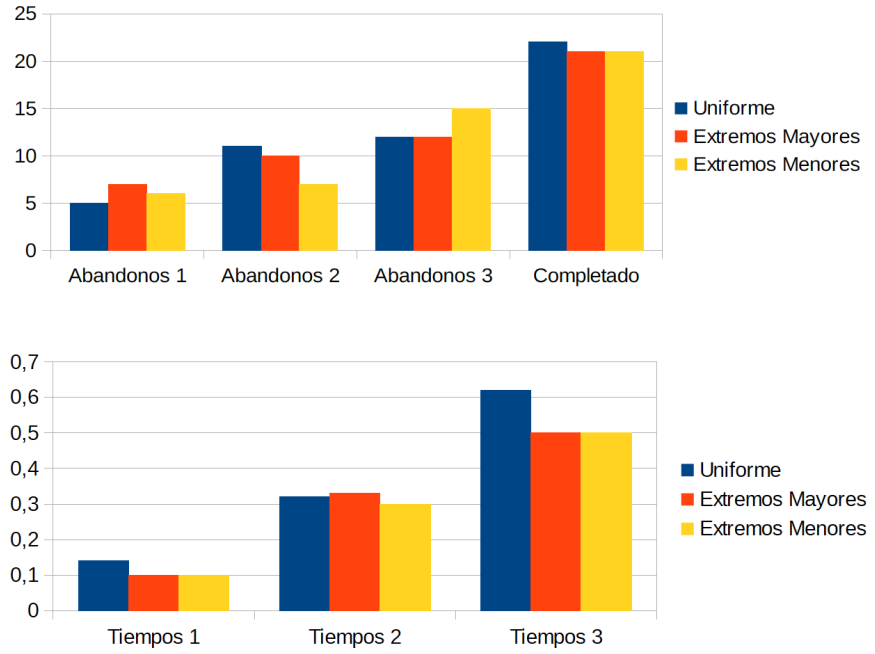

Figura 4.2: Resultados Simulación A

De las resultados mostrados en las gráficas [Véase en Figura 4.2], se puede extraer las siguientes conclusiones:

- El tiempo empleado en cada actividad está directamente relacionado con la dificultad de esta.
- El alto porcentaje de alumnos con motivación alta ha provocado que un gran número de alumnos completen la estrategia con éxito.
- Los abandonos producidos en cada actividad están relacionados con la dificultad de las actividades, aunque se puede observar que los resultados han sido mas equitativos que en la gráfica de tiempos.

Escenario B

Datos:

- Población : 50 alumnos
- Alumnos con motivación baja : 10 %
- Alumnos con motivación media : 80 %
- Alumnos con motivación alta : 10 %
- Dificultad Actividad 1 : baja
- Dificultad Actividad 2 : media
- Dificultad Actividad 3 : alta

Resultados:

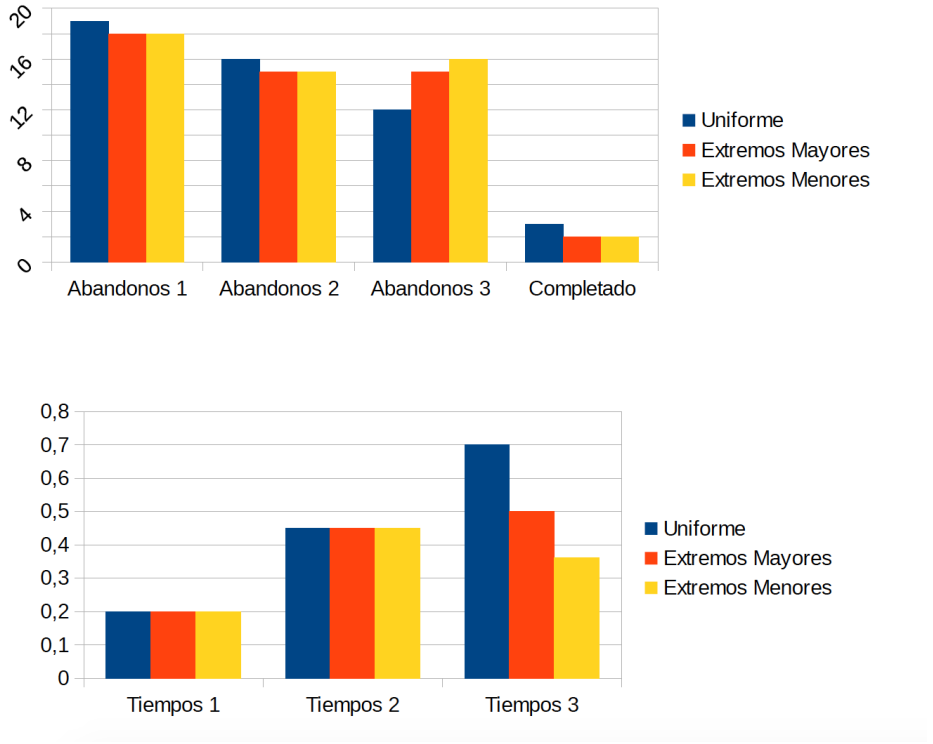

Figura 4.3: Resultados Simulación B

De las resultados mostrados en las gráficas [Véase en Figura 4.3], se puede extraer las siguientes conclusiones:

- El tiempo empleado en cada actividad sigue estando directamente relacionado con la dificultad de cada una.
- El bajo porcentaje de alumnos con motivación alta ha provocado que un gran número de alumnos no completen la estrategia con éxito.
- Los abandonos producidos en cada actividad están muy parejos, además son mucho mas altos que en la ejecución anterior. De esto se puede deducir como conclusión que los abandonos están muy relacionados con la motivación de los alumnos aparte de estarlo con la dificultad de las actividades.

Escenario C

Datos:

- Población : 50 alumnos
- Alumnos con motivación baja : 80 %
- Alumnos con motivación media : 10 %

- Alumnos con motivación alta : 10 %
- Dificultad Actividad 1 : baja
- Dificultad Actividad 2 : media
- Dificultad Actividad 3 : alta

Resultados:

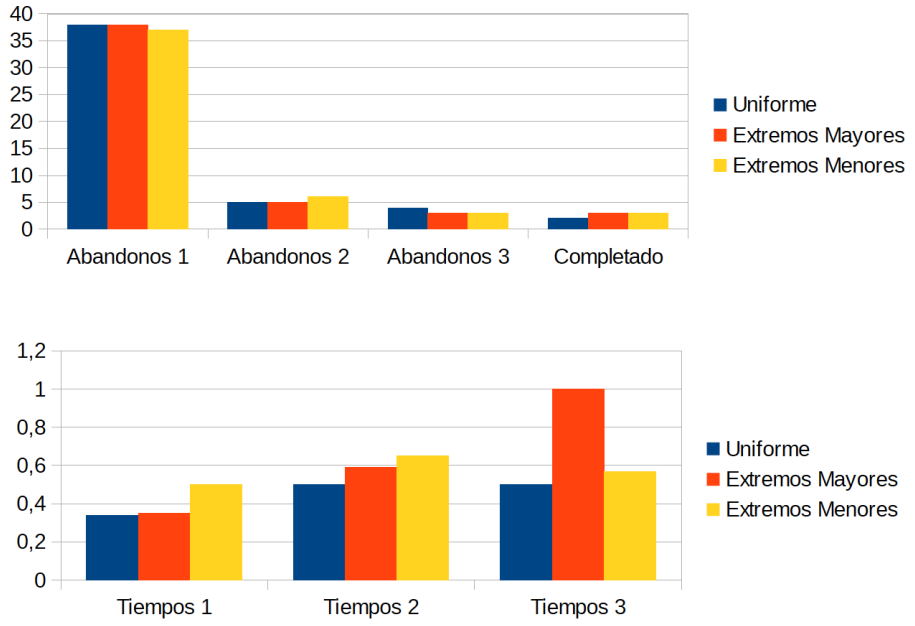

Figura 4.4: Resultados Simulación C

De las resultados mostrados en las gráficas [Véase en Figura 4.4], se puede extraer las siguientes conclusiones:

- El tiempo empleado en cada actividad sigue estando directamente relacionado con la dificultad de cada una.
- El alto porcentaje de alumnos con motivación baja ha provocado que un gran número de alumnos no completen la estrategia con éxito, y además abandonen la estrategia en la primera actividad.
- La motivación de los alumnos tiene más peso que la dificultad de las actividades a la hora de su resolución o no.

Conclusiones

Después de observar los resultados del experimento, se valida que se cumple el comportamiento esperado de los alumnos. Para ello, el modelo de simulación cumple los siguientes requisitos:

- El tiempo empleado en realizar las actividades depende directamente de la dificultad de éstas. Este comportamiento era uno de los objetivos principales ya que se acerca bastante a la vida real.
- Los abandonos se guían por la tasa de éxito [Véase en Cuadro 4.5]. La idea es que un alumno realizase con éxito o no una actividad dependiendo de la tasa o porcentaje de éxito que reciba. Este porcentaje viene asignado en el Cuadro 4.5 y será el que el usuario crea conveniente.

  El éxito de un alumno depende de su motivación y de la dificultad de la actividad a realizar, aquí destacamos que se ha dado más peso o importancia a la motivación del alumno que a la dificultad de las actividades, ya que en el mundo real la motivación tiene más fuerza que las dificultades que nos podamos encontrar.

Capítulo 5

Análisis del Sistema

En este capítulo, se describirán todas las actividades de análisis realizadas durante todas las iteraciones del proyecto, análisis del sistema, los requisitos de información, funcionales y no funcionales, el catálogo de actores, los diagramas de casos de uso, la especificación de cada caso de uso, diagramas de secuencia y reglas de negocio. También se hablará sobre el estudio de alternativas tecnológicas.

5.1. Requisitos de información

A continuación, en los Cuadros 5.1, 5.2, 5.3, 5.4, 5.5, 5.6 y 5.7 se describen los requisitos de información.

IRQ-001	PremioWS	
Descripción	Representa los premios que se compartirán en SmartWeb	
Atributos		
Nombre	**Tipo**	**Descripción**
nombre	Texto	Nombre del premio
valor	Numérico	Valor del premio
descripción	Texto	Descripción del premio

Cuadro 5.1: Requisitos de Información: IRQ-001

IRQ-002	Premio	
Descripción	Representa los premios asignados a las actividades de la estrategia.	
Atributos		
Nombre	**Tipo**	**Descripción**
id	Texto	Identificador del premio
valor	Texto	Valor del Premio
criterio	Texto	Criterio para asignar el premio.

Cuadro 5.2: Requisitos de Información: IRQ-002

IRQ-003	Criterio	
Descripción	Representa una toma de decisión mediante un criterio	
Atributos		
Nombre	**Tipo**	**Descripción**
criterioID	Texto	Identificador del criterio
criterioValue	Texto	Valor para este criterio
targetActivityId	Texto	Actividad destino

Cuadro 5.3: Requisitos de Información: IRQ-003

IRQ-004	Coordenada	
Descripción	Representa una coordenada	
Atributos		
Nombre	**Tipo**	**Descripción**
x	Texto	Valor para coordenada x
y	Texto	Valor para coordenada y

Cuadro 5.4: Requisitos de Información: IRQ-004

IRQ-005	Actividad	
Descripción	Representa una actividad de la estrategia	
Atributos		
Nombre	**Tipo**	**Descripción**
activityTypeId	Texto	Id de la actividad
activityTitle	Texto	Nombre de la actividad
hijo	Texto	Id de la actividad hija de esta
dificultad	Numérico	dificultad de la actividad
criterios	Lista	Criterios asignados a esta actividad
premios	Lista	Premios asignados a esta actividad
x	Texto	Valor para coordenada x
y	Texto	Valor para coordenada y

Cuadro 5.5: Requisitos de Información: IRQ-005

IRQ-006	Estudiante	
Descripción	Representa a un estudiante del curso	
Atributos		
Nombre	**Tipo**	**Descripción**
id	Texto	Identificador del estudiante en BD
nif	Texto	NIF del estudiante
username	Texto	Nombre de usuario de Moodle
badges	Numérico	Número de medallas
first	Numérico	Número de puntos First-Time
points	Numérico	Número de puntos
xp	Numérico	Número de puntos de Experiencia
level	Numérico	Nivel del alumno

Cuadro 5.6: Requisitos de Información: IRQ-006

IRQ-007	Estrategia	
Descripción	Estrategia de gamificación.	
Atributos		
Nombre	**Tipo**	**Descripción**
strategyId	Cadena	Identificador de la estrategia
userId	Cadena	Usuario que creó la estrategia
strategyTitle	Cadena	Nombre de la estrategia
courseId	Cadena	Identificador del curso
nAlumnos	Cadena	Número de alumnos para la simulación
motivacionBaja	Numérico	Porcentaje de alumnos con la motivación baja
motivacionMedia	Numérico	Porcentaje de alumnos con la motivación media
motivacionAlta	Numérico	Porcentaje de alumnos con la motivación alta
distribucion	Numérico	Indica que tipo de distribución tiene la simulación
duracion	Cadena	Duración de la simulación
rutaExcel	Cadena	Ruta del Excel con la tasa de éxito
actividades	Lista	Actividades de la estrategia

Cuadro 5.7: Requisitos de Información: IRQ-007

5.2. Requisitos funcionales

En este apartado se definen los requisitos funcionales, que describen las distintas funcionalidades que el sistema será capaz de realizar. Véase en Cuadros 5.8, 5.9, 5.10, 5.11, 5.12 y 5.13.

Código	FRQ-001
Nombre	Abrir Estrategias
Descripción	El usuario podrá acceder a las estrategias almacenadas en un archivo XML.
Iteración	Iteración 1
Trazabilidad	SUBOBJ-002, SUBOBJ-003, SUBOBJ-004

Cuadro 5.8: Requisitos Funcionales: FRQ-001

Código	FRQ-002
Nombre	Cerrar Estrategias
Descripción	El usuario podrá cerrar las estrategias de gamificación abiertas con anterioridad.
Iteración	Iteración 1
Trazabilidad	-

Cuadro 5.9: Requisitos Funcionales: FRQ-002

Código	FRQ-003
Nombre	Generar Archivo de Simulación
Descripción	El usuario podrá generar un archivo de simulación.
Iteración	Iteración 2
Trazabilidad	SUBOBJ-002

Cuadro 5.10: Requisitos Funcionales: FRQ-003

Código	FRQ-004
Nombre	Abrir archivo de simulación en Anylogic
Descripción	El usuario podrá abrir el archivo generado en Anylogic de manera automática desde la aplicación.
Iteración	Iteración 2
Trazabilidad	SUBOBJ-002

Cuadro 5.11: Requisitos Funcionales: FRQ-004

Código	FRQ-005
Nombre	Aplicar Estrategia a Moodle
Descripción	El usuario podrá aplicar las estrategias de gamificación en Moodle.
Iteración	Iteración 3
Trazabilidad	SUBOBJ-003

Cuadro 5.12: Requisitos Funcionales: FRQ-005

Código	FRQ-006
Nombre	Compartir resultados en la red social SmartWeb
Descripción	El usuario podrá compartir los resultados obtenidos de aplicar la estrategia de gamificación en base de datos.
Iteración	Iteración 3
Trazabilidad	SUBOBJ-003, SUBOBJ-004

Cuadro 5.13: Requisitos Funcionales: FRQ-006

5.2.1. Catálogo de Actores

A continuación, se especifica el catálogo de actores de la aplicación. En este caso, sólo hay un tipo de actor que será cualquier usuario de la aplicación [Cuadro 5.14].

Código	ACT-001
Nombre	Usuarios
Descripción	Todos los usuarios de la aplicación.

Cuadro 5.14: Catálogo de Actores: ACT-001

5.2.2. Diagramas de Casos de Uso

En la Figura 5.1 se describe el diagrama de casos de usos.

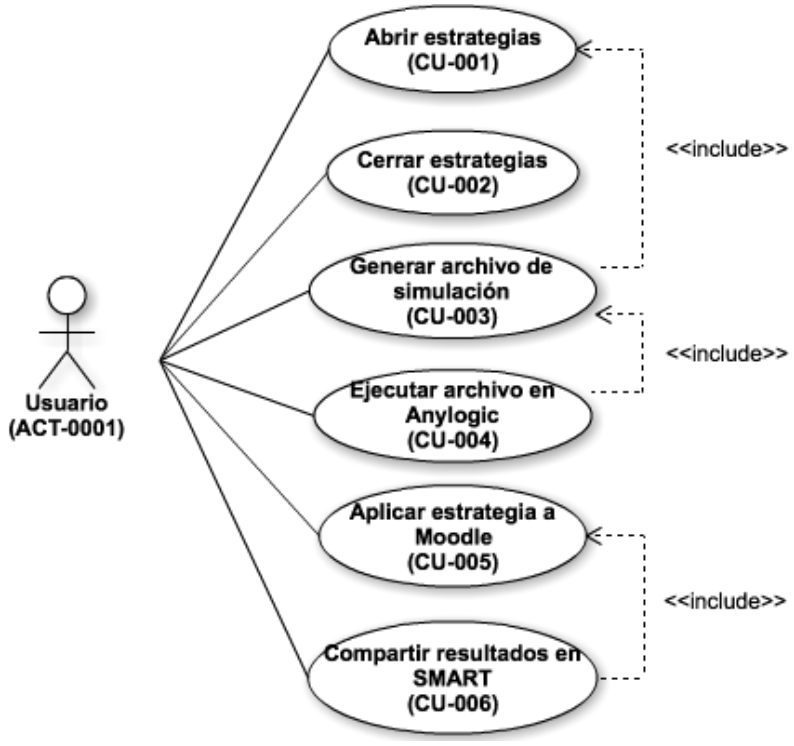

Figura 5.1: Diagrama Casos de Uso

5.2.3. Especificación de Casos de Uso

A continuación, en los Cuadros 5.15, 5.16, 5.17, 5.18, 5.19 y 5.20 se especificarán los diferentes casos de uso del sistema.

Código	CU-001
Nombre	Abrir Estrategias
Dependencia	FRQ-001
Descripción	El usuario abrirá el archivo XML que contiene las distintas estrategias.
Precondición	-
Postcondición	Se muestran las distintas estrategias.
Secuencia Normal	
Paso	**Acción**
1	El usuario pulsa el botón Open.
2	La aplicación muestra una pantalla para que el usuario seleccione el archivo a abrir.
3	El usuario selecciona el archivo.
4	la aplicación abre el fichero y lee el contenido con las estrategias almacenadas.
5	La aplicación muestra en la sección Strategies las distintas estrategias cargadas.
Excepciones	
Paso	**Acción**
3	El usuario selecciona un archivo no compatible con la aplicación.
4	La aplicación muestra un mensaje de error y termina el caso de uso.

Cuadro 5.15: Casos de Uso: CU-001

Código	CU-002
Nombre	Cerrar Estrategias
Dependencia	FRQ-002
Descripción	El usuario podrá cerrar las estrategias cargadas en la aplicación.
Precondición	Se ha realizado CU-001.
Postcondición	La sección Strategies aparece vacía.
Secuencia Normal	
Paso	**Acción**
1	El usuario pulsa el botón Close.
2	La aplicación muestra una ventana de confirmación.
3	El usuario confirma la acción.
4	La aplicación cierra las estrategias que estaban abiertas.
Excepciones	
Paso	**Acción**
3	El usuario cancela la acción y termina el caso de uso.

Cuadro 5.16: Casos de Uso: CU-002

Código	CU-003
Nombre	Generar archivo de simulación
Dependencia	FRQ-003
Descripción	El usuario generará un archivo de simulación ejecutable en Anylogic.
Precondición	Se ha realizado CU-001.
Postcondición	Se genera el archivo y se vacía el formulario.
Secuencia Normal	
Paso	**Acción**
1	El usuario selecciona una estrategia.
2	El usuario pulsa el botón Simulation.
3	La aplicación muestra una pantalla con un formulario.
4	El usuario rellena el formulario y pulsa el botón Generate.
5	La aplicación muestra una pantalla para que el usuario elija dónde guardar el nuevo archivo.
6	El usuario selecciona dónde guardar y guarda.
7	Se guarda el archivo generado.
8	La aplicación muestra un mensaje de que la acción se ha realizado correctamente.
Excepciones	
Paso	**Acción**
8	La aplicación muestra un mensaje de error porque el archivo no se ha generado correctamente.

Cuadro 5.17: Casos de Uso: CU-003

Código	CU-004
Nombre	Abrir archivo de simulación en Anylogic
Dependencia	FRQ-004
Descripción	El usuario abre el archivo generado por la aplicación en Anylogic.
Precondición	Se ha realizado CU-003.
Postcondición	Se abre el archivo generado en Anylogic.
Secuencia Normal	
Paso	**Acción**
1	El usuario pulsa el botón Run Anylogic.
2	Se abre Anylogic con el modelo de simulación generado para la estrategia seleccionada por el usuario.
Excepciones	
Paso	**Acción**
2	La aplicación muestra un mensaje de error porque no se ha podido abrir Anylogic.

Cuadro 5.18: Casos de Uso: CU-004

Código	CU-005
Nombre	Aplicar estrategia a Moodle
Dependencia	FRQ-005
Descripción	El usuario aplica a Moodle la estrategia seleccionada.
Precondición	Se ha realizado CU-001.
Postcondición	Se muestran los resultados.
Secuencia Normal	
Paso	**Acción**
1	El usuario selecciona una estrategia.
2	El usuario pulsa el botón Moodle
3	La aplicación muestra la pantalla correspondiente a esta opción.
4	El usuario pulsa el botón Apply to Moodle
5	La aplicación consulta la base de datos de Moodle para obtener los datos necesarios para aplicar la estrategia.
6	La aplicación muestra los resultados obtenidos.
Excepciones	
Paso	**Acción**
6	La aplicación muestra un mensaje de error porque ha habido algún problema para conectar con Moodle y termina el caso de uso.

Cuadro 5.19: Casos de Uso: CU-005

Código	CU-006
Nombre	Compartir resultados en la red social SmartWeb
Dependencia	FRQ-006
Descripción	El usuario comparte en la red social SmartWeb los resultados de aplicar la estrategia a Moodle.
Precondición	Se he realizado CU-004.
Postcondición	Se guardan los datos.
Secuencia Normal	
Paso	**Acción**
1	El usuario pulsa el botón Share
2	La aplicación mediante servicios web se comunica con SmartWeb para compartir los resultados.
3	La aplicación muestra un mensaje de que la operación se ha realizado con éxito.
Excepciones	
Paso	**Acción**
3	La aplicación muestra un mensaje de error porque no se ha podido realizar la operación y termina el caso de uso.

Cuadro 5.20: Casos de Uso: CU-006

5.2.4. Diagramas de Secuencia

En esta sección se muestran los diagramas de secuencia de los casos de uso de la aplicación: Abrir estrategias [Véase en Figura 5.2], cerrar estrategias [Véase en Figura 5.3], generar archivo de simulación [Véase en Figura 5.5], ejecutar simulación en Anylogic [Véase en Figura 5.4], aplicar estrategia a Moodle [Véase en Figura 5.6] y Compartir los resultados en SmartWeb [Véase en Figura 5.7].

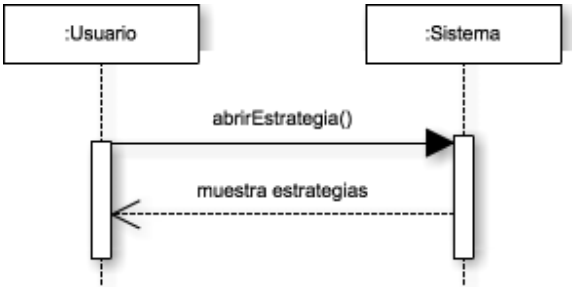

Figura 5.2: Diagrama de Secuencia CU-001

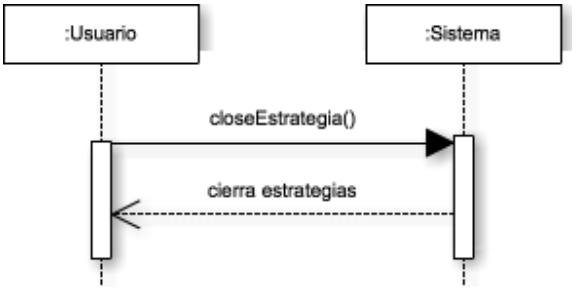

Figura 5.3: Diagrama de Secuencia CU-002

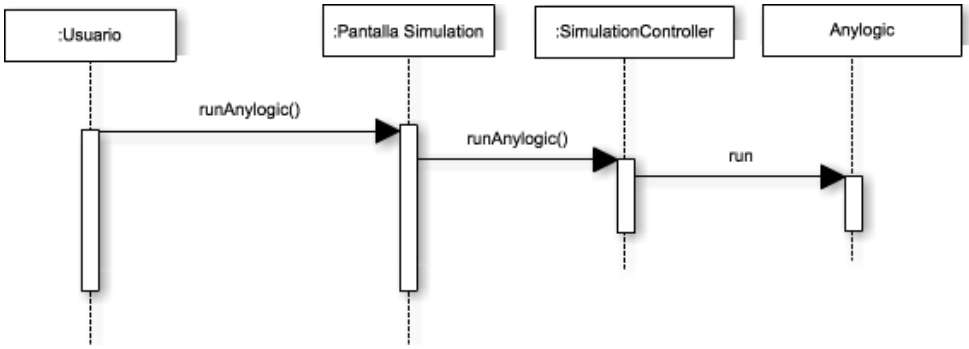

Figura 5.4: Diagrama de Secuencia CU-004

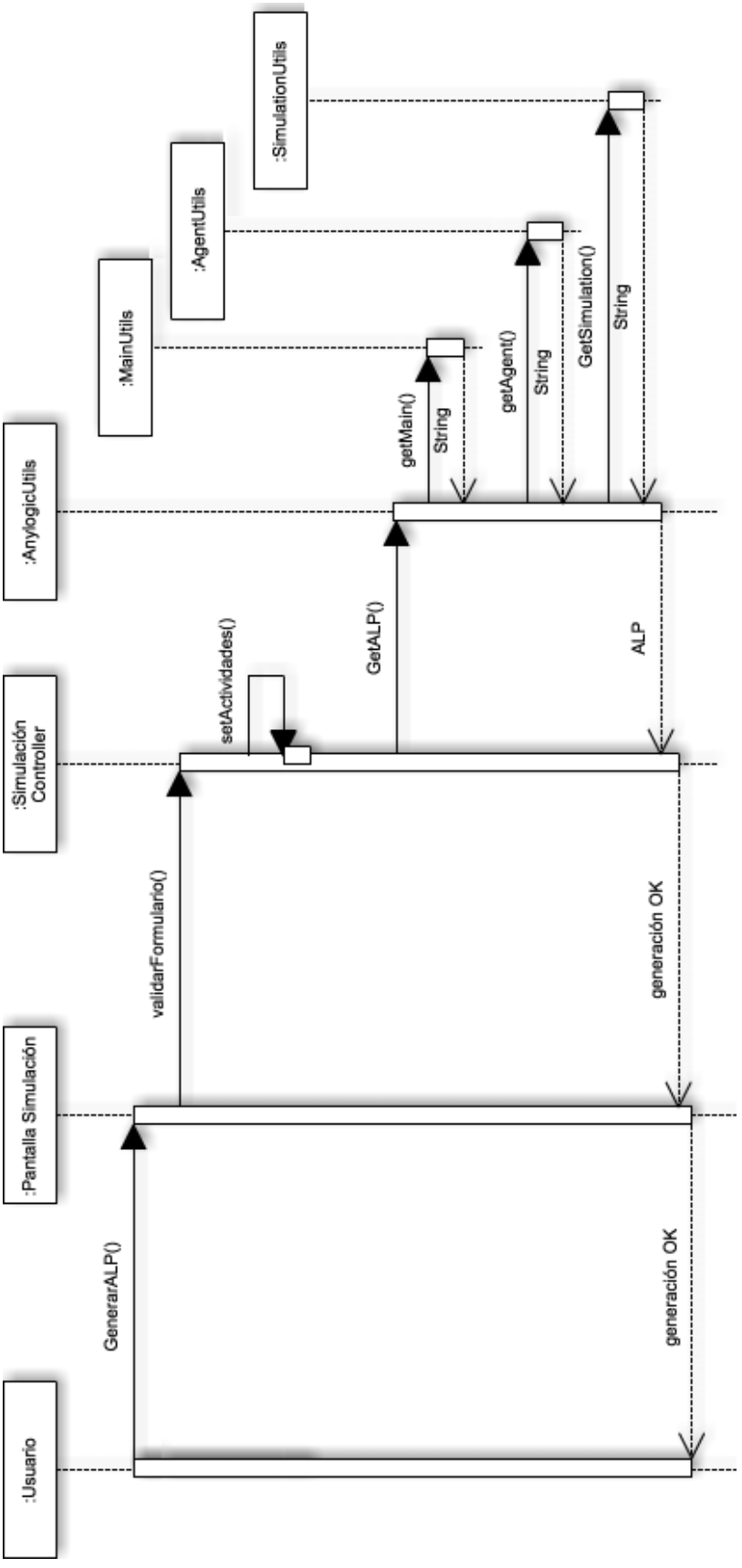

Figura 5.5: Diagrama de Secuencia CU-003

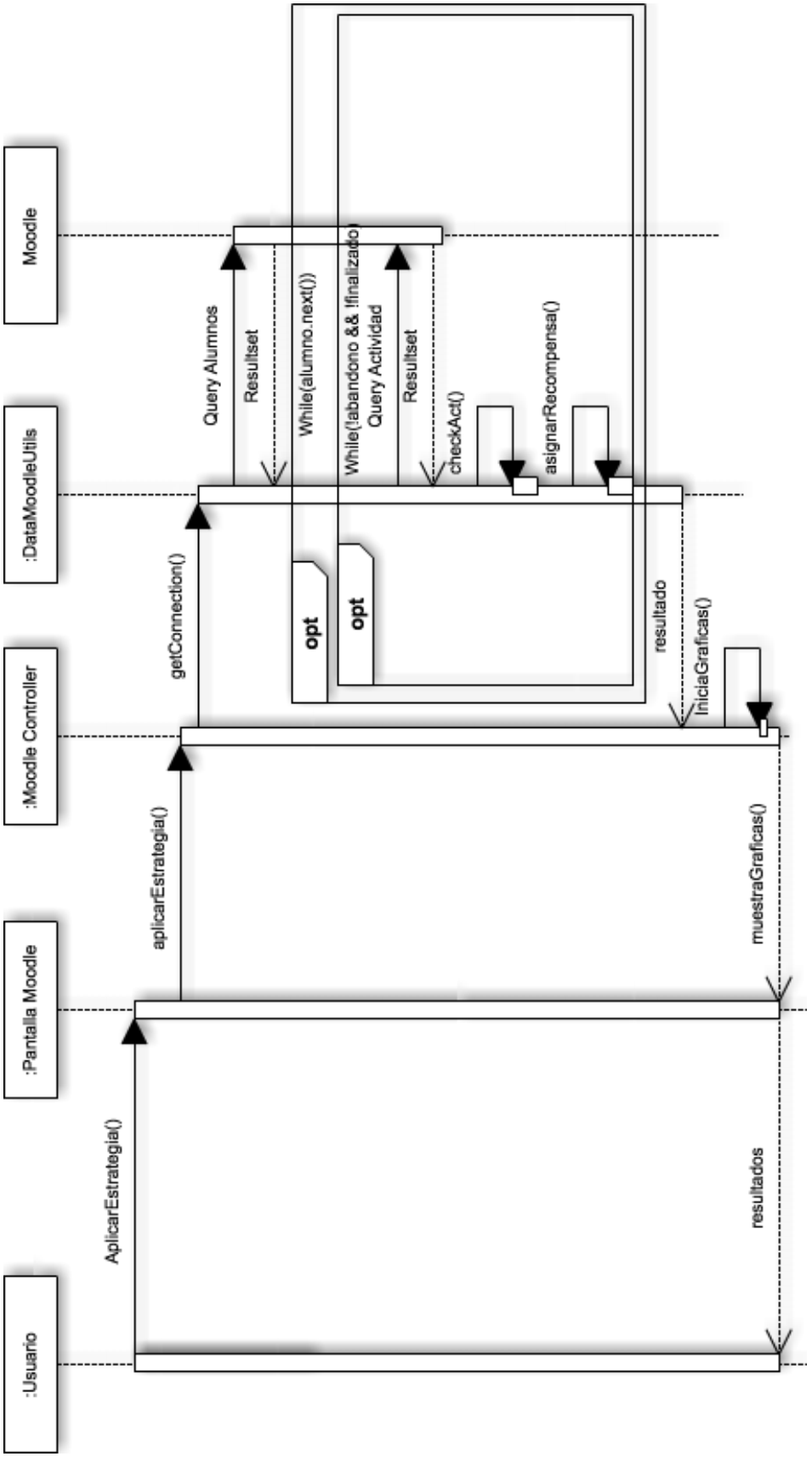

Figura 5.6: Diagrama de Secuencia CU-005

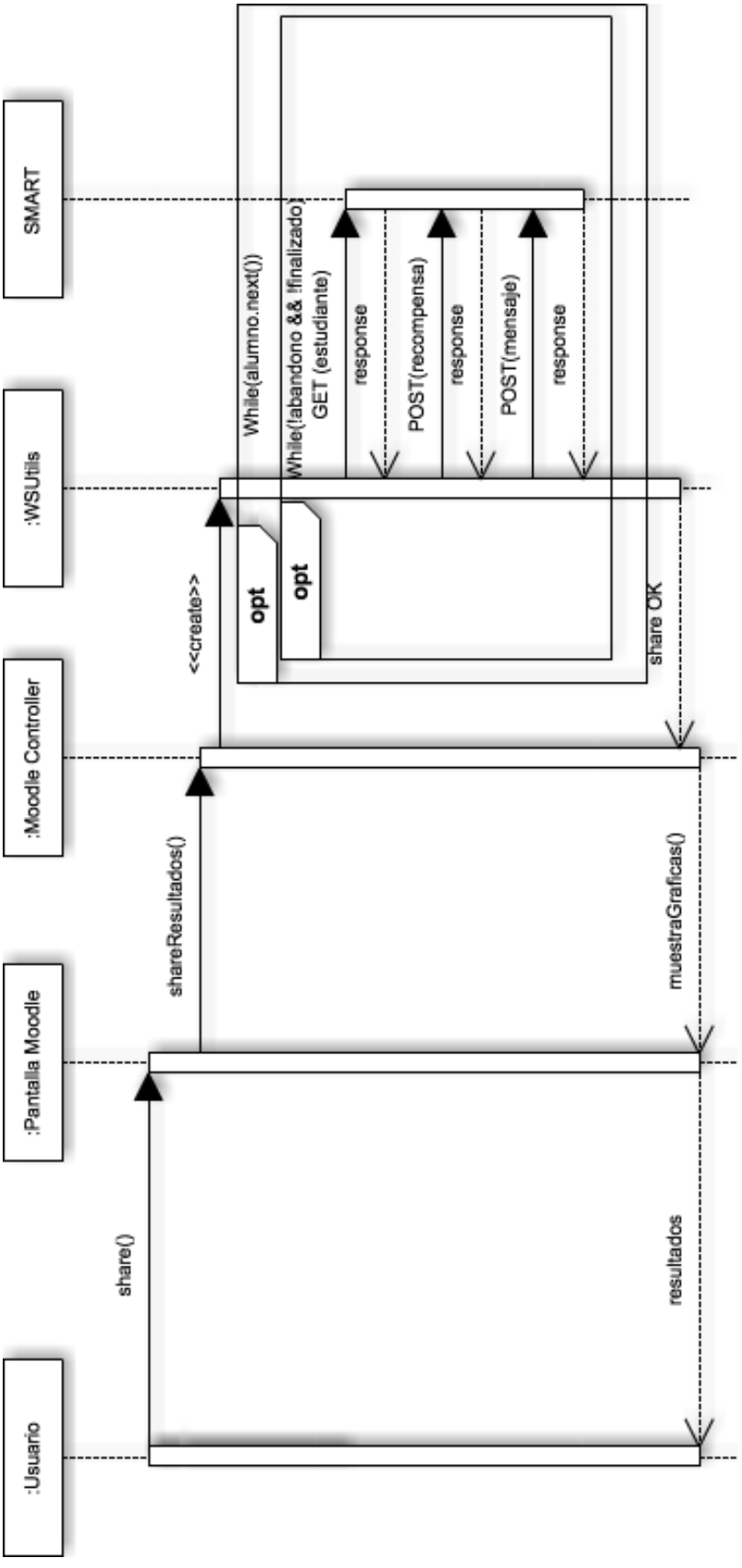

Figura 5.7: Diagrama de Secuencia CU-006

5.3. Requisitos no funcionales

A continuación, en los Cuadros 5.21, 5.22, 5.23, 5.24, 5.25 y 5.27 se describen los requisitos no funcionales.

Código	NRQ-001
Nombre	Entorno
Descripción	La aplicación debe funcionar en el sistema operativo Windows y Mac OS.

Cuadro 5.21: Requisitos no Funcionales: NRQ-001

Código	NRQ-002
Nombre	Idioma
Descripción	El contenido de la aplicación debe estar en inglés.

Cuadro 5.22: Requisitos no Funcionales: NRQ-002

Código	NRQ-003
Nombre	Usabilidad
Descripción	Facilidad de aprendizaje y de uso.

Cuadro 5.23: Requisitos no Funcionales: NRQ-003

Código	NRQ-004
Nombre	Compatibilidad
Descripción	El archivo de simulación generado debe ser ejecutable en la aplicación Anylogic.

Cuadro 5.24: Requisitos no Funcionales: NRQ-004

Código	NRQ-005
Nombre	Rendimiento
Descripción	A aplicación debe tener unos tiempos de respuesta lo más bajo posibles.

Cuadro 5.25: Requisitos no Funcionales: NRQ-005

Código	NRQ-006
Nombre	Mantenibilidad
Descripción	La aplicación debe ofrecer facilidad de mantenimiento, y facilidad de actualización hacia versiones más modernas.

Cuadro 5.26: Requisitos no Funcionales: NRQ-006

5.4. Reglas de negocio

Este apartado se describirán las restricciones y reglas de negocio de la aplicación. Algunas de las restricciones que se describen a continuación se deben a limitaciones de algunos de los sistemas en uso, otras son simples reglas establecidas por el desarrollador relacionadas con la lógica del sistema.

- Una estrategia no puede contener mas de 24 actividades, esta restricción se debe a una limitación de Anylogic. Al ejecutar el archivo de simulación en Anylogic se abre una pantalla con un espacio determinado, en esta pantalla solo cabe un determinado número de componentes. La cantidad elegida es la justa para que la interfaz no resulte sobrecargada y provoque problemas de solapamientos de componentes. Esta restricción es controlada por la aplicación.
- El número de alumnos que participarán en la simulación es introducido por el usuario. Deberá ser un número entero mayor que 0. Esta restricción es controlada por la aplicación.
- Los porcentajes de alumnos con las distintas motivaciones son introducidos por el usuario. Deberán ser números enteros entre 0 y 100. Esta restricción es controlada por la aplicación.
- La suma de los tres porcentajes debe ser 100. Esta restricción es controlada por la aplicación.
- En la pantalla de simulación es obligatorio rellenar todos los campos para poder generar el archivo correspondiente. Esta restricción es controlada por la

aplicación.

- Es necesario tener instalado el programa Anylogic en su versión PLE, para poder ejecutar el archivo de simulación.
- SmartWeb debe estar activo y disponible en el servidor.

5.5. Estudio de alternativas tecnológicas

En este apartado de describirán las distintas alternativas tecnológicas para el desarrollo de esta aplicación.

Lenguaje de Programación

Dada la dificultad de la parte de simulación, se decidió utilizar un lenguaje de programación conocido por parte del desarrollador, esto permitiría un gran ahorro de tiempo. El lenguaje elegido fue Java. Con este lenguaje se eligió entre dos librerías diferentes:

- Java Swing, un framework MVC para desarrollar interfaces gráficas para Java. Algunas de sus características son: independencia de plataforma, su extensibilidad, es muy personalizable.
- JavaFX, un framework MVC para desarrollar interfaces gráficas para Java. Las aplicaciones JavaFX pueden ser ejecutadas en una amplia variedad de dispositivos. Permite crear aplicaciones de escritorio, para móviles, aplicaciones Web, consolas de videojuegos y reproductores Blu-ray entre otros. Una de sus principales características es la posibilidad de dar estilo a la interfaz con CSS.

IDE

Como IDE de desarrollo se estudió el uso de Eclipse y Netbeans.

- Eclipse, es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para el desarrollo de aplicaciones. Una de sus principales características es que provee al programador con frameworks muy potentes para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de software, aplicaciones web, etc.
- NetBeans, es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, web, EJB y aplicaciones móviles). Entre sus características se encuentra un sistema de proyectos basado en Ant, control de versiones y refactoring.

Conexión con base de datos Moodle

El SGBD de Moodle es MySQL, por lo que solo había una alternativa. Se ha utilizado el driver JDBC (Java Database Connectivity) para la conexión con la base de datos.

Conexión con red social SmartWeb

Esta aplicación solo es accesible mediante servicios web Rest, los datos se envían en formato JSON. Para ello se han utilizado librerías Jersey y Gson.

5.6. Mapeo Requisitos-Iteraciones

En el cuadro 5.27 se muestra el mapeo de los requisitos establecidos en cada iteración.

Iteración	**Requisitos**
Iteración 0	NRQ-001, NRQ-002, NRQ-003, NRQ-004, NRQ-005 y NRQ-006
Iteración 1	IRQ-002, IRQ-003, IRQ-005, IRQ-007, FRQ-001 y FRQ-002
Iteración 2	IRQ-004, FRQ-003 y FRQ-004
Iteración 3	IRQ-001, IRQ-006, FRQ-005 y FRQ-006

Cuadro 5.27: Mapeo Requisitos-Iteraciones

Capítulo 6

Diseño

6.1. Arquitectura del Sistema

A continuación, se describirá todas las tareas de diseño realizadas durante todas las iteraciones, entre ellas se pueden destacar la arquitectura física y lógica del sistema desarrollado, así como el patrón de diseño utilizado para el desarrollo.

6.1.1. Arquitectura física

En este apartado, se detalla la arquitectura física de la aplicación [Figura 6.1]. El sistema recibe de entrada un archivo XML que contiene una serie de estrategias de gamificación. Una vez que el sistema carga esta información, se dispone de dos subsistemas diferentes:

- Simulación: Este subsistema se encarga de generar un archivo de simulación con extensión .ALP, de la estrategia previamente seleccionada por el usuario, este archivo es ejecutable en el software de simulación Anylogic.
- Moodle: Este subsistema tiene una arquitectura Cliente-Servidor. La parte cliente es este subsistema de la propia aplicación. En la parte servidor disponemos de dos servidores distintos. Por un lado, tenemos la instalación de Moodle, y por el otro tenemos la red social SmartWeb.

  La interacción con Moodle se basa en lanzar consultas a su base de datos para obtener la información necesaria. Estar consultas se realizan por medio de un driver JDBC .

  La interacción con la red social SmartWeb se basa en compartir los logros obtenidos por los alumnos. La comunicación con SmartWeb se realiza por medio de servicios web REST, y la información que se envía y se recibe tiene un formato JSON.

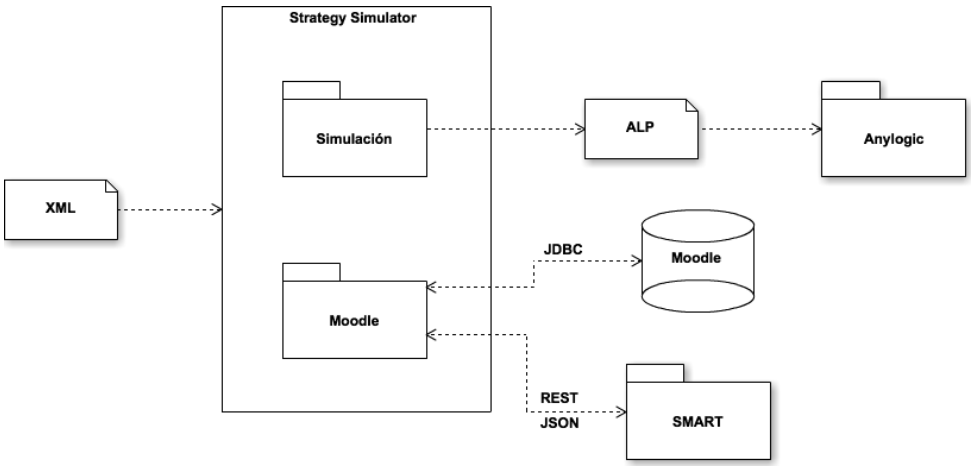

Figura 6.1: Arquitectura Física

6.1.2. Arquitectura lógica

La arquitectura lógica del sistema está formada por cada uno de los artefactos software que la componen. La arquitectura lógica se encuentra dividida en dos bloques, componentes desarrollados y componentes usados [Figura 6.2].

Componentes Desarrollados

- **Módulo Simulación:** Este módulo conforma toda la lógica necesaria para la generación de archivo de simulación. Desde la lectura de los parámetros introducidos por el usuario, hasta la construcción del archivo ALP.
- **Módulo Moodle:** Este módulo contiene de la lógica para el procesamiento de los datos recogidos de Moodle, así como el procesamiento de datos que posteriormente se publicarán en la red social SmartWeb.
- **AnylogicUtils:** Es una librería desarrollada para la generación de los componentes que conforman la simulación en Anylogic.
- **DataMoodleUtils:** Es una librería desarrollada para las utilidades relacionadas con Moodle, desde la creación de la conexión, hasta la creación de las consultas realizadas a base de datos.
- **WSUtils:** Esta librería se ha desarrollado para que contenga todos los métodos necesarios para la utilización de los servicios web Rest, necesarios para la comunicación de la aplicación con la red social SmartWeb.

Componentes Utilizados

- **Anylogic:** Software de simulación utilizado para simular el modelo generado por la aplicación.
- **Moodle:** Plataforma para el aprendizaje, utilizada por la Universidad de Cá-

diz y en la cual se quiere realizar un estudio para su gamificación.

- **SmartWeb:** Red social de ámbito docente, en la que se publicarán los logros obtenidos por los usuarios.

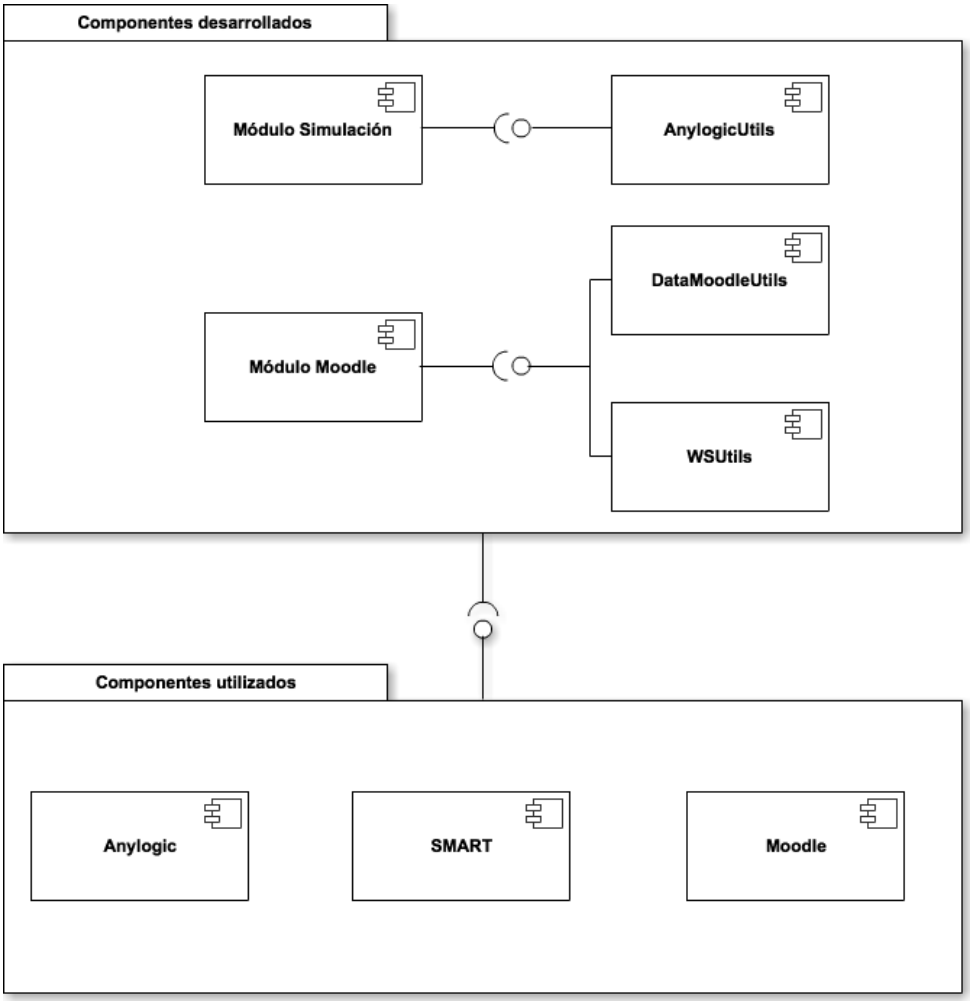

Figura 6.2: Arquitectura Lógica

6.1.3. Patrón de diseño

La arquitectura de diseño especifica la forma en que los artefactos software, interactúan entre sí para lograr el comportamiento deseado en el sistema. La aplicación desarrollada se basa en el patrón de diseño Modelo-Vista-Controlador [Figura 6.3].

Este patrón se basa en la reutilización de código y la separación de la vista, la lógica de negocio y el modelo. Estas características ofrecen calidad y claridad en el código y facilitar el mantenimiento del sistema.

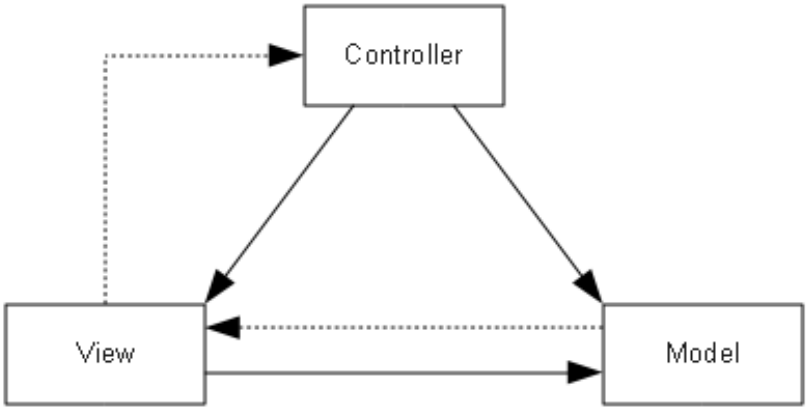

Figura 6.3: Patrón de diseño MVC

- **Modelo**, representa la información de la aplicación. En nuestra aplicación no es necesaria la persistencia de datos por lo que, la comunicación con la base de datos se utiliza para la obtención de los datos necesarios para aplicar la estrategia a Moodle.
- **Vista**, representa la capa que está en contacto con el usuario. En esta aplicación la interfaz se ha creado con componentes de la librería JavaFx, y el estilo mediante CSS.
- **Controlador**, recibe, interpreta y da respuesta a los eventos enviados tanto por el usuario como por la propia aplicación. Debe interactuar con el modelo y con la vista. En esta aplicación el controlador contendrá código Java

6.2. Modelo de dominio

Para el funcionamiento de sistema no se requiere la persistencia de datos, por lo que no hay un modelo de datos que diseñar. Se ha usado el modelo de datos de Moodle, necesario para aplicar las estrategias de gamificación, aunque debido a su tamaño y número de tablas no se puede mostrar su esquema de base de datos. En el Capítulo 7 se describirá con mas detalle. A continuación, se describirá el modelo de dominio del sistema [Figura 6.4], que representa cada una de las entidades utilizadas para el funcionamiento de la aplicación.

- **Estrategia**, entidad que representa a una estrategia de gamificación. Una estrategia contiene una o muchas actividades.
- **Actividad**, entidad que representa a una actividad de las que forman una estrategia. Una actividad puede tener asignado cero o varios premios, como puede tener cero o varios criterios asociados a ella.
- **Premio**, entidad que representa a un premio asociado a una actividad.

- **Estudiante**, entidad que representa a un estudiante.
- **PremioWS**, entidad que representa a un premio que será publicado en la red social SmartWeb.
- **Criterio**, entidad que representa a un criterio para la evaluación de una actividad.
- **Coordenada**, entidad que representa una coordenada en un plano (X,Y). Estas coordenadas son necesarias para colocar en una determinada posición cada componente de Anylogic.

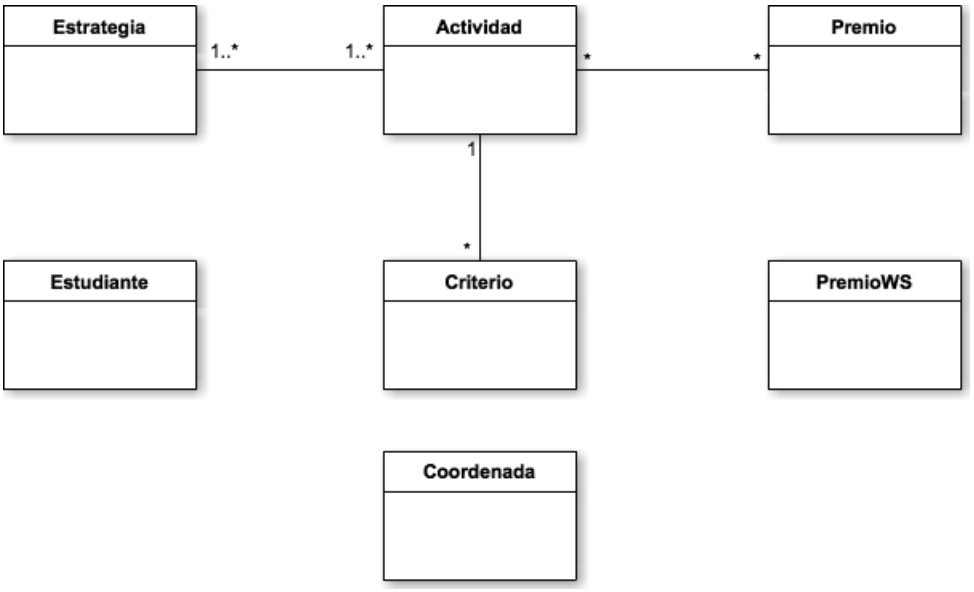

Figura 6.4: Modelo de dominio

6.3. Diseño de la interfaz de usuario

La interfaz de usuario hace referencia al medio que permite a un usuario comunicarse con la máquina. Para esta aplicación se han diseñado dos interfaces de usuario. La primera interfaz es la de la propia aplicación, una interfaz bastante sencilla, clara e intuitiva para el usuario. La segunda interfaz es para el archivo de simulación ejecutable en Anylogic. Anylogic no ofrece una gran flexibilidad para el diseño, sin embargo, se ha intentado crear una interfaz sencilla y lo más atractiva posible para el usuario.

Interfaz Strategy Simulator

En la Figura 6.5 se puede observar la navegabilidad que tiene la interfaz de usuario de nuestra aplicación.

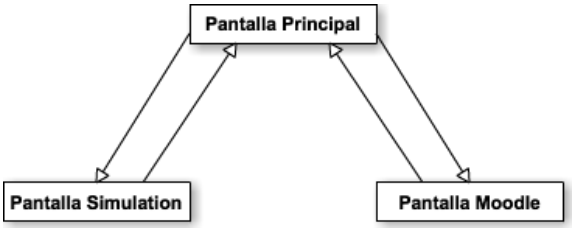

Figura 6.5: Navegación de la aplicación

- **Pantalla principal**, es la pantalla inicial de la aplicación. Desde ella se puede acceder a cualquiera de las dos funcionalidades. También cuenta con el panel de mandos general de la aplicación desde el cual se puede cargar estrategias en la aplicación, borrar las estrategias cargadas, acceder aun diálogo con información sobre la aplicación y salir de ésta [Véase en Figura 6.6].
- **Pantalla Anylogic**, en esta pantalla se encuentra el formulario para que el usuario introduzca los parámetros para generar la simulación. También es posible abrir el archivo generado en la aplicación Anylogic. Además contiene el panel general de la aplicación descrito en el apartado de pantalla principal [Véase en Figura 6.7].
- **Pantalla Moodle**, esta nos permite aplicar la estrategia a Moodle, y obtener unas gráficas con los datos obtenidos. También nos da la posibilidad de compartir estos datos en la red social SmartWeb. Además contiene el panel general de la aplicación descrito en el apartado de pantalla principal [Véase en Figura 6.8].

Figura 6.6: Pantalla Principal

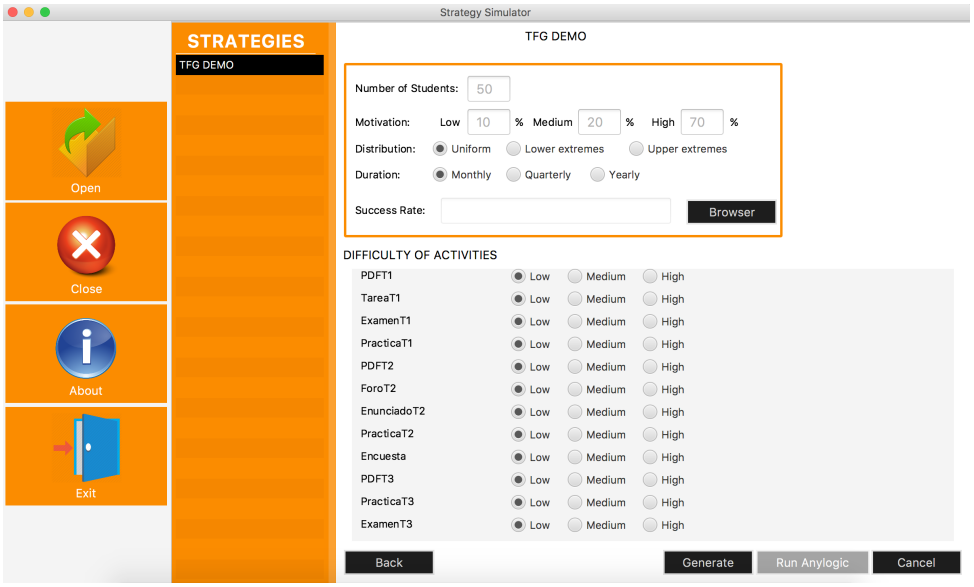

Figura 6.7: Pantalla de Simulación

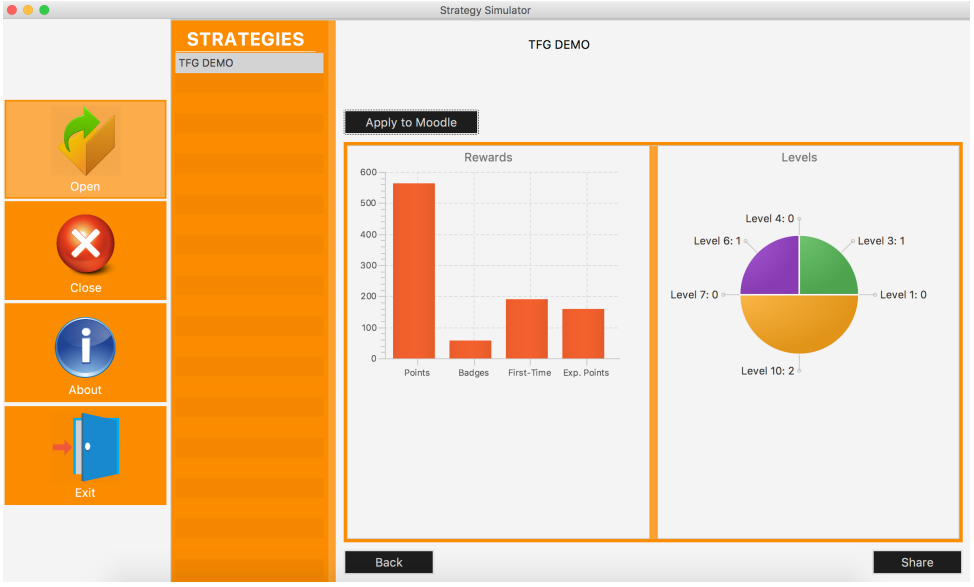

Figura 6.8: Pantalla de Moodle

Interfaz en Anylogic

En la Figura 6.9 se puede observar la navegabilidad que tiene la interfaz de Anylogic.

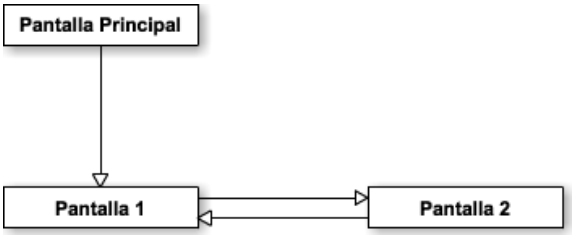

Figura 6.9: Navegación del archivo ejecutable

- **Pantalla Principal**, en esta pantalla se encuentra el formulario con los parámetros introducidos por el usuario en la aplicación, en esta pantalla es posible modificar estos parámetros sin necesidad de tener que generar el archivo de nuevo. También contiene el botón RUN que arranca la simulación [Véase en Figura 6.10].
- **Pantalla 1**, esta pantalla contiene dos histogramas con resultados de la simulación. El primer histograma muestra el número de alumnos que han abandonado cada actividad de la estrategia. El segundo indica el tiempo medio empleado por los alumnos en la realización de cada actividad [Véase en Figura 6.11].
- **Pantalla 2**, esta pantalla contiene dos gráficos de sectores con resultados de la simulación. El primero muestra el número premios conseguidos por los alumnos del curso, estos logros se encuentran clasificados por tipo de premio. El segundo indica el número de alumnos que han acabado la estrategia en cada nivel. Por último también muestra el numero de alumnos que han completado la estrategia al completo [Véase en Figura 6.12].

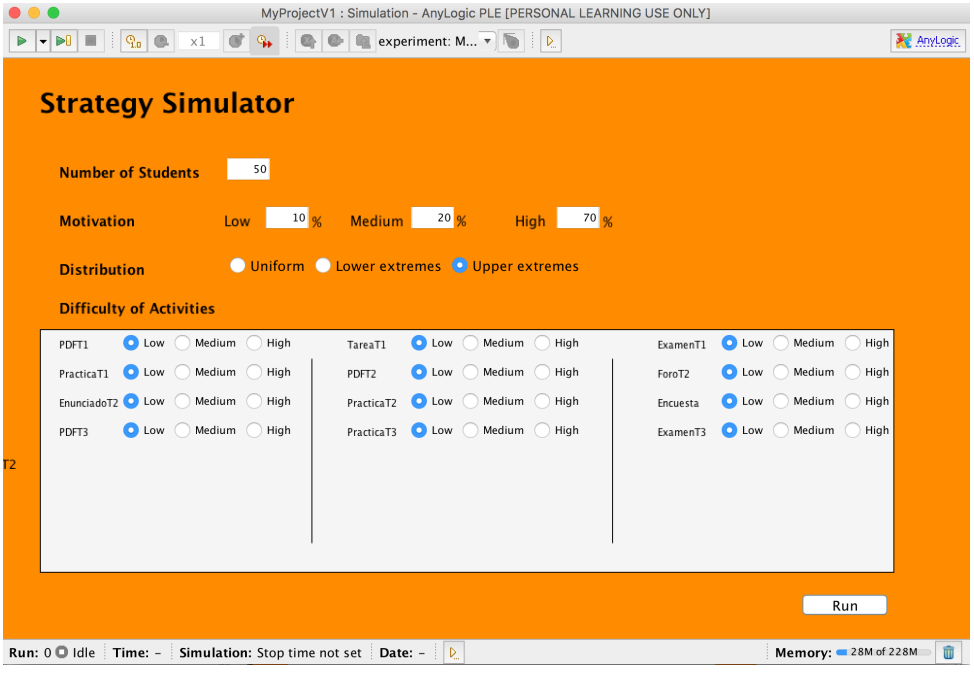

Figura 6.10: Pantalla Principal Anylogic

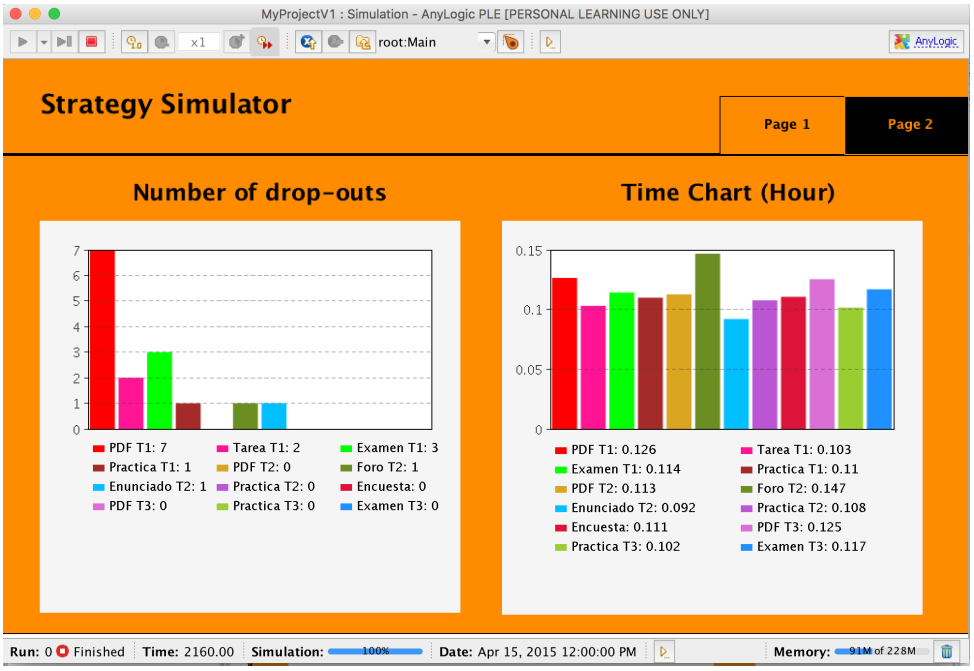

Figura 6.11: Pantalla de resultados 1

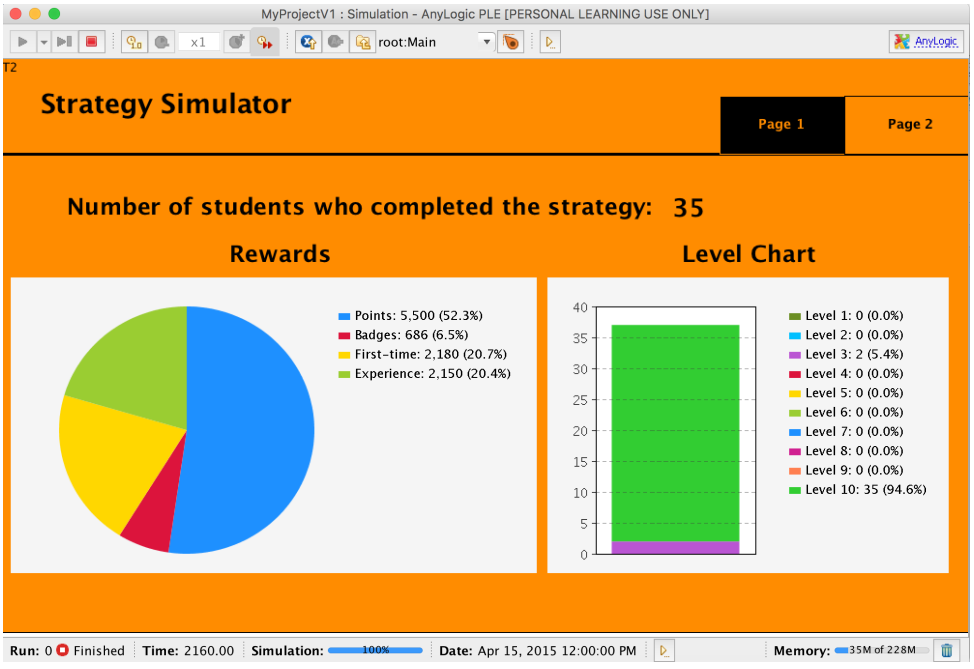

Figura 6.12: Pantalla de resultados 2

6.4. Mapeo Diseño-Iteraciones

En el cuadro 6.1 se muestra el mapeo de los componentes diseñados en cada iteración.

Iteración	**Diseño**
Iteración 0	Primer diseño del modelo de simulación y arquitectura de la aplicación.
Iteración 1	Primer diseño de la interfaz de usuario.
Iteración 2	Diseño final de la interfaz general de la aplicación y diseño final de la pantalla de simulación.
Iteración 3	Diseño final de la pantalla de Moodle.

Cuadro 6.1: Mapeo Diseño-Iteraciones

Capítulo 7

Desarrollo del Sistema

En este capítulo se describen las distintas herramientas y tecnologías utilizadas durante el desarrollo de este proyecto. También se describen todos los componentes implementados durante todas las iteraciones del desarrollo del proyecto.

7.1. Entorno de desarrollo

Como entorno de desarrollo se ha utilizado Eclipse, se ha elegido esta herramienta por ser bastante conocida por el desarrollador, lo que ha implicado no tener que dedicar un tiempo adicional en el aprendizaje de una herramienta de desarrollo.

Eclipse

Eclipse es un entorno de desarrollo integrado de código abierto multiplataforma para hacer lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones de "Clienteliviano" basadas en navegadores. Esta plataforma se ha usado para crear entornos de desarrollo integrados (IDE), como el IDE de Java, llamado Java Development Toolkit (JDT) y el compilador (ECJ) que se entrega como parte de Eclipse y que son usados también para desarrollar el mismo Eclipse.

Fue creado originalmente por IBM como el sucesor de VisualAge. Actualmente, es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios. Se liberó originalmente bajo la Common Public License, pero después fue re-licenciado bajo la Eclipse Public License; sin embargo, la Free Software Foundation ha determinado que ambas licencias, aún siendo software libre, son incompatibles con la Licencia pública general de GNU (GNU GPL). El IDE de Eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma de cliente enriquecido, a diferencia de los entornos monolíticos, donde todas las funcionalidades están incluidas, tanto si les son útiles al usuario como si no. Con este sistema de módulos, el usuario tiene un entorno ligero, personalizado según sus necesidades. Con estos módulos, podemos extender la funcionalidad de Eclipse a otros

lenguajes de programación.

7.1.1. Lógica de negocio

En la implementación de la lógica de negocio se han utilizado las siguientes tecnologías:

JAVA

Java es un lenguaje de programación orientado a objetos que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo, lo que quiere decir que el código que es ejecutado en una plataforma no tiene que ser recompilado para ejecutarse en otra. Java es, a partir de 2012, uno de los lenguajes de programación más populares en uso, particularmente para aplicaciones web cliente-servidor.

Para esta aplicación en concreto se ha utilizado la biblioteca JavaFx que hace que el desarrollo de aplicaciones de escritorio complejas y con gran dinamismo, usabilidad, etc. sea relativamente sencillo.

Se ha elegido este lenguaje de programación porque es conocido por el desarrollador de la aplicación y por tener un gran uso para el desarrollo de aplicaciones de escritorio.

XML

XML, siglas en inglés de eXtensible Markup Language, es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el World Wide Web Consortium (W3C) utilizado para almacenar datos en forma legible. Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

XML no ha nacido sólo para su aplicación para Internet, sino que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo y casi cualquier cosa imaginable.

Esta tecnología se ha utilizado para almacenar las estrategias de gamificación. Esto permite una gran flexibilidad a la hora de construir y almacenar las estrategias, cosa que, por ejemplo, en bases de datos relacionales no se tiene. A continuación, se puede observar la estructura del XML.

```
<strategies>
  <strategy>
    <strategyId>1</strategyId>
    <strategyTitle>TFG DEMO</strategyTitle>
    <strategyUserId>Daniel</strategyUserId>
    <originalUserId>Daniel Guerrero</originalUserId>
    <courseId>4</courseId>
    <activities>
      <activity>
        <activityId>2004</activityId>
        <activityTypeId>2</activityTypeId>
        <activityTitle>Practica T3</activityTitle>
        <targetActivityID/>
        <decission>
          <criterios>
            <criterio>
              <criterioID>333</criterioID>
              <valueCriterio>7</valueCriterio>
              <targetActivityID/>
            </criterio>
        </decission>
        <rewards>
            <reward>
              <valueCriterio>10</valueCriterio>
              <rewardID>5</rewardID>
            </reward>
        </rewards>
      </activity>
    </activities>
  </strategy>
</strategies>
```

En este ejemplo se puede observar la estructura que tendría una estrategia de ejemplo. La estrategia tiene los siguientes atributos:

- strategyId, es el identificador de la estrategia.
- strategyTitle, nombre de la estrategia.
- strategyUserId, usuario de la estrategia.
- originalUserId, usuario que diseñó la estrategia.
- courseId, es el identificador del curso de Moodle, al que se aplica la estrategia.
- activities, conjunto de actividades que componen la estrategia.

Cada actividad tiene los siguientes atributos:

- activityId, es el identificador de la actividad.

- activityTypeId, identificador del tipo de actividad. Una actividad puede ser de varios tipos como pueden ser: Recurso, Tarea, Cuestionario, etc.
- activityTitle, nombre de la actividad.
- targetActivityID, actividad que le sigue en la estrategia. Si este campo está vacío significa que es la ultima actividad de la estrategia o que hay alguna toma de decisión.
- decission, indica si dependiendo del resultado de la actividad hay una toma de decisión.
- rewards, premios o recompensas asignadas a esta actividad.

Una decisión se compone de criterios. Estos criterios sirven para evaluar el resultado de la actividad y realizar una toma de decisión:

- criterioID, es el identificador del criterio.
- valueCriterio, es el valor que tiene que tener el resultado de la actividad para cumplir este criterio.
- targetActivityID, actividad que le sigue en la estrategia si cumple este criterio. Una actividad puede tener un criterio que apunte a ella misma, esto quiere decir que, si cumple el criterio tiene que repetir la actividad. Si este campo está vacío significa que es la ultima actividad de la estrategia.

Una recompensa contiene los siguientes atributos:

- rewardID, es el tipo de recompensa. Véase en Capítulo 4
- valueCriterio, es el valor de la recompensa.

7.1.2. Capa de persistencia

MySQL

MySQL es un sistema de gestión de base de datos relacional. MySQL AB desarrolla este sistema como software libre en un esquema de licencia dual, siendo GNU GPL excepto para aquellas empresas que quieran incorporarlo en productos privativos. Es probablemente el más popular en su género, usado por sitios web como Google, Facebook, Twitter y YouTube. Es la opción escogida para interactuar con la plataforma Moodle.

Esta tecnología es la que ofrece Moodle como SGBD, por ello, su uso. Debido al tamaño de la base de datos de Moodle no se puede explicar todos los componentes usado por ello vamos explicar uno de los ejemplos que se usa en este proyecto.

A continuación, se muestra un ejemplo del uso del modelo de datos de Moodle:

Por cada alumno del curso se va comprobando si ha realizado cada una de las actividades de la estrategia. Para ello se realiza una consulta de base de datos por

cada actividad. En Moodle cada tipo de actividad se almacena en una tabla diferente, es decir, hay que hacer consultas diferentes. A continuación, se describirá un ejemplo de como se evalúa si un alumno ha superado o no la entrega de una tarea.

Para evaluar una tarea debemos comprobar que su nota supera el criterio, por lo que debemos consultar en base de datos la nota que tiene asociada.

Las tareas se almacenan en la tabla **mdl_assign** y la nota asociada a cada tarea se almacenan en la tabla **mdl_assign_grades**. Estas tablas tienen el siguiente esquema [Figura 7.1].

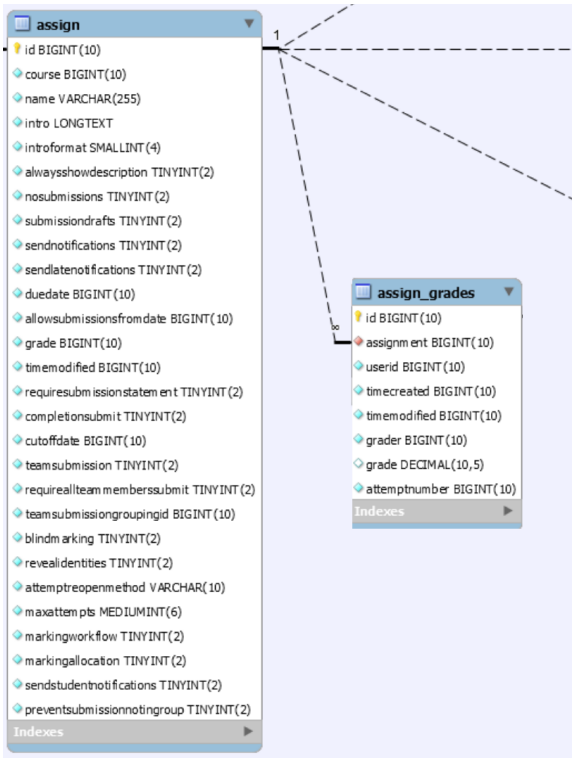

Figura 7.1: Esquema Tablas: mdl_assign y mdl_assign_grades

Como se puede observar la tabla contiene mucha información. Para el caso de las tareas solo utilizaremos los siguientes atributos:

Tabla mdl_assign

- id, identificador de la tarea.
- course, identificador del curso al que pertenece la tarea.

Tabla mdl_assign_grades

- grade, nota de la tarea.
- assignment, identificador de la tarea evaluada.
- userid, identificador del usuario al que se evalúa.

Para comprobar la nota del alumno realizamos la siguiente consulta:

```
SELECT mdl_assign_grades.grade
FROM mdl_assign_grades, mdl_assign
WHERE mdl_assign.id = mdl_assign_grades.assignment
AND mdl_assign.course = CURSO
AND mdl_assign_grades.userid = USERID
AND mdl_assign.id = TAREAID
```

Donde CURSO es el identificador del curso que vamos a consultar, USERID es el identificador del alumnos que vamos a consultar y TAREAID es el identificador de la tarea que vamos a comprobar.

Esta consulta nos devuelve la nota que este usuario tiene asociada a esta tarea. Si no devuelve nada significa que esta tarea no ha sido evaluada, por lo que significaría que no ha superado la actividad. En el caso de que devuelva algún valor se comprueba si es mayor o igual al criterio para superar la actividad, establecido por el diseñador de la estrategia de gamificación. En el caso que este criterio esté vacío, se establece por defecto que una tarea es superada si su nota es mayor o igual a 5.

Servicios Web REST

REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web. El término se originó en el año 2000, en una tesis doctoral sobre la web escrita por Roy Fielding, uno de los principales autores de la especificación del protocolo HTTP y ha pasado a ser ampliamente utilizado por la comunidad de desarrollo.

Esta tecnología es la que usa SmartWeb para la persistencia de datos, por ello, su uso.

Principales características:

- Un protocolo cliente/servidor sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.
- Un conjunto de operaciones bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son POST, GET, PUT y DELETE. Con frecuencia, estas operaciones se equiparan a las operaciones CRUD en bases de datos (crear, leer, actualizar, borrar) que se requieren para la persistencia de datos.
- Una sintaxis universal para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- El uso de hipermedios, tanto para la información de la aplicación como para las transiciones de estado de la aplicación: la representación de este estado en un sistema REST son típicamente HTML o XML. Como resultado de esto, es posible navegar de un recurso REST a muchos otros, simplemente siguiendo enlaces sin requerir el uso de registros u otra infraestructura adicional.

JSON

JSON, acrónimo de JavaScript Object Notation, es un formato de texto ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript aunque hoy, debido a su amplia adopción como alternativa a XML, se considera un formato de lenguaje independiente.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON.

Esta tecnología es la que usa SmartWeb para el intercambio de datos.

7.1.3. Interfaz de usuario

CSS

CSS son las siglas de Cascading Style Sheets, u hojas de estilo en cascada en castellano. Se trata de un lenguaje de hojas de estilo usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. Complementa al documento HTML dando forma al contenido.

FXML

FXML es un lenguaje basado en XML y utilizado para definir interfaces en JavaFX. Aunque java JavaFX permite crear interfaces mediante su código, al usar FXML es mucho mas sencillo adaptar las ventanas y paneles a los diferentes tamaños necesarios. Es ideal para diseñar cualquier interfaz de usuario ya que es compatible con la clase Scene de JavaFX.

JavaFX Scene Builder

Con la aplicación JavaFX Scene Builder podemos construir la interfaz gráfica de una aplicación de escritorio Java de forma más sencilla. JavaFX Scene Builder genera archivos descriptores FXML que podemos cargar en la aplicación evitando la tediosa y no sencilla tarea de construir la interfaz gráfica mediante código.

7.2. Código fuente

En esta sección se describe la organización del código fuente, describiendo la utilidad de los diferentes ficheros y su distribución en paquetes o directorios.

tfg.myproject

Este paquete contiene la clase principal de la aplicación:

- **MainApp.java:** Clase que contiene la lógica principal de la aplicación.

tfg.myproject.model

Este paquete contiene las clases que representan los objetos utilizados en la aplicación. También se utiliza como modelo para el tratamiento de datos.

- **Actividad.java:** Esta clase representa a las distintas actividades que componen la estrategia de gamificación.
- **Coordenada.java:** Esta clase representa una coordenada (X,Y).
- **Criterio.java:** Esta clase representa el criterio que se va a seguir para conceder una recompensa a un alumno.
- **Estrategia.java:** Esta clase representa la estrategia de gamificación que usará la aplicación.
- **Estudiante.java:** Esta clase representa a los estudiantes de Moodle cuya información se va a usar.
- **Premio.java:** Esta clase representa las recompensas que recibirán los estudiantes al realizar las actividades.
- **PremioWS.java:** Esta clase representa los premios que se van a compartir en la red social SmartWeb.

tfg.myproject.view:

Este paquete contiene los archivos que componen la interfaz de usuario, así como las clases controladoras.

- **StrategyOverviewController.java:** Clase controladora principal de la aplicación. También se encarga de la navegación entre pantallas.
- **MenuBarController.java:** Clase controladora del panel de control común a toda la aplicación.
- **AboutController.java:** Clase controladora del diálogo de información sobre la aplicación.
- **ErrorController.java:** Clase controladora del diálogo de error.
- **InfoController.java:** Clase controladora del diálogo de información.
- **StrategyOverview.fxml:** Interfaz principal de la aplicación. Se compone de la pantalla principal, la pantalla de simulación y la pantalla de Moodle.
- **RootLayout.fxml:** Interfaz del panel de control común a toda la aplicación.
- **InfoDialog.fxml:** Interfaz del diálogo de información.
- **ErrorDialog.fxml:** Interfaz del diálogo de error.
- **AboutView.fxml:** Interfaz del diálogo que contiene información sobre la aplicación.

tfg.myproject.util

Este paquete contiene clases con distintas utilidades para conseguir el funcionamiento

de las distintas funcionalidades de la aplicación.

- **AnylogicUtils.java:** Esta clase contiene la lógica para construir un archivo ALP.
- **AgentUtils.java:** Esta clase contiene la lógica para construir la parte agente del archivo ALP.
- **SimulationUtils.java:** Esta clase contiene la lógica para construir la parte simulation del archivo ALP.
- **MainUtils.java:** Esta clase contiene la lógica para construir la parte main del archivo ALP.
- **Data.java:** Esta clase contiene la lógica necesaria para establecer la conexión con la base de datos de Moodle.
- **DataMoodleUtils.java:** Esta clase contiene la lógica necesaria para realizar las consultas a la base de datos de Moodle.
- **Config.java:** Esta clase contiene la lógica para leer del archivo db.properties la configuración de la conexión a base de datos mediante JDBC.
- **WSUtils.java:** Esta clase contiene la lógica necesaria para realizar las llamadas a los servicios REST.

7.3. Otras herramientas

7.3.1. Subversion

Subversion es un sistema de control de versiones, diseñado para reemplazar a CVS. Es un software libre con licencia Apache/BSD. También se le conoce por svn, ya que este es el nombre que usa en la línea de órdenes. Una de las principales diferencias entre Subversion y CVS es que en el primero todo el proyecto está bajo el mismo número de revisión; mientras que en el segundo cada archivo tiene un número de revisión independiente.
El hecho de que se pueda acceder al repositorio a través de la red permite que varias personas trabajen sobre el mismo repositorio. Esto fomenta la colaboración, y la calidad no se resiente, ya que si algún cambio es incorrecto, siempre se puede volver a una versión anterior a dicho cambio. El uso de esta herramienta ha aportado a este proyecto las siguientes ventajas:

- Se ha podido mantener un historial de los archivos y directorios [Figura 7.2].
- Ha permitido la creación de ramas, muy útil para que nuevos desarrollos no afecten de manera negativa a otros ya realizados y probados.
- Gracias a que registra los cambios por subidas al repositorio, y no por cambios en cada archivo, ha permitido recuperar del repositorio si ha sido necesario

copias de seguridad de funcionalidades completas.

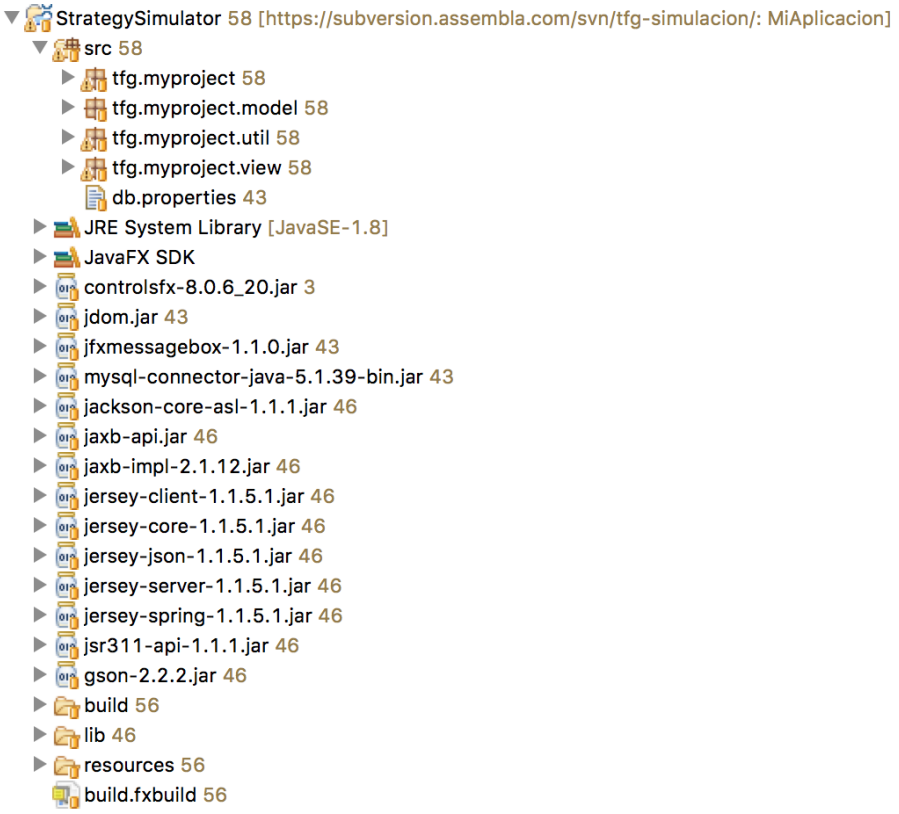

Figura 7.2: Estructura del proyecto bajo control de versiones

7.3.2. Assembla

Assembla es una empresa que provee herramientas de colaboración y de seguimiento de errores y tareas basadas en la nube para organizar y administrar proyectos de código abierto y comerciales para el desarrollo de software. Actualmente, Assembla tiene más de 500.000 usuarios en más de 100 países. Esta herramienta se ha utilizado como repositorio para esta aplicación.

7.3.3. Trello

Trello es un software de administración de proyectos con interfaz web, para organizar tareas de manera colaborativa [Figura 7.3].

Surge en 2010 como un proyecto de la compañía Fog Creek Software. En 2011 sale la primera versión como una aplicación web y cliente para iPhone. En 2012 se lanza la versión para android. Para 2014 se crea Trello Inc. separándose de Fog.

Empleando el sistema kanban, para el registro de actividades con tarjetas virtuales organiza tareas, permite agregar listas, adjuntar archivos, etiquetar eventos, agregar

comentarios y compartir tableros.

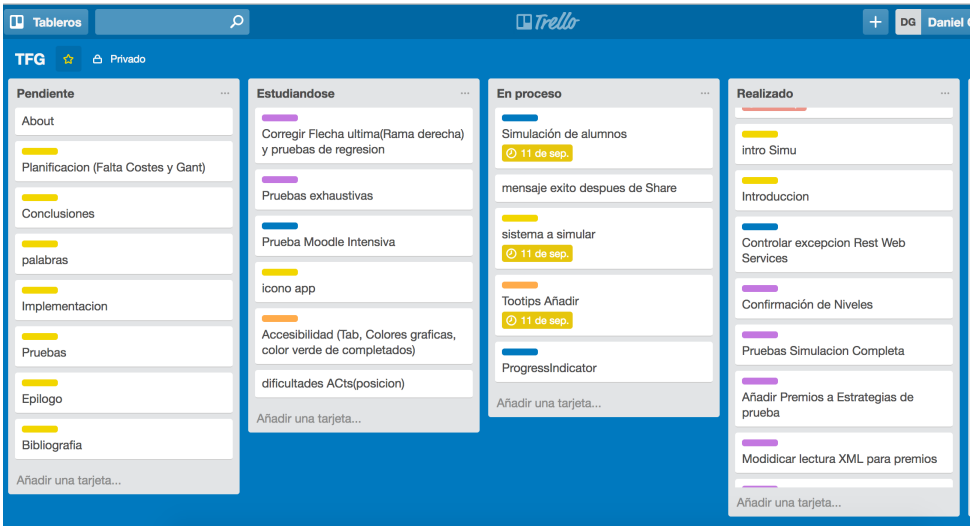

Figura 7.3: Herramienta Trello

7.3.4. MAMP

El acrónimo MAMP se refiere al conjunto de programas software comúnmente usados para desarrollar sitios web dinámicos sobre sistemas operativos Apple Macintosh, MAC OS X [Figura 7.4].

- Mac OS X: Sistema operativo.
- Apache: Servidor Web.
- MySQL: Sistema Gestor de Bases de Datos.
- PHP, Perl ó Python, lenguajes de programación usados para la creación de sitios web.

Este software se ha utilizado para la instalación local de la plataforma Moodle.

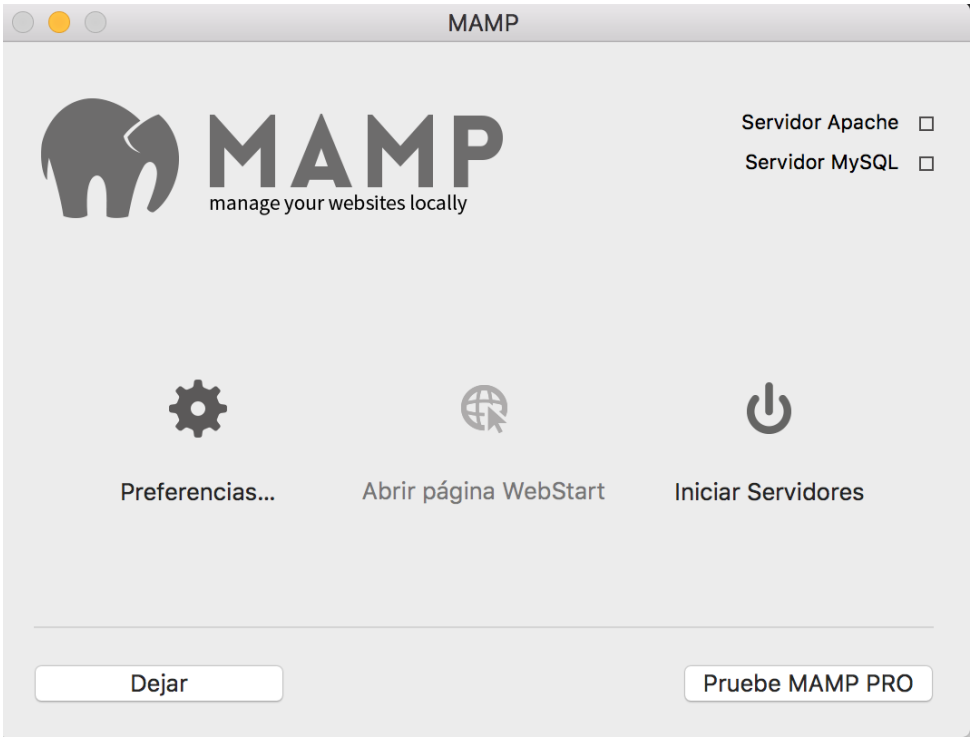

Figura 7.4: MAMP

7.3.5. Cacoo

Cacoo es una herramienta 2.0 de dibujo y diseño en línea amigable que permite crear una gran variedad de diagramas como mapas de sitios, wireframes, diagramas de UML y de red, con la particularidad de poder hacerlos de forma colaborativa con otros usuarios en tiempo real y cuenta con un chat para poder hablar y comunicarse con los usuarios durante estos trabajos colaborativos [Figura 7.5].

Para utilizarlo, como es habitual, se puede registrar de forma gratuita usando las cuentas de Twitter, Facebook o Google, así como realizar un registro normal con email y contraseña. Existe también una versión de pago, que aumenta las prestaciones de la aplicación.

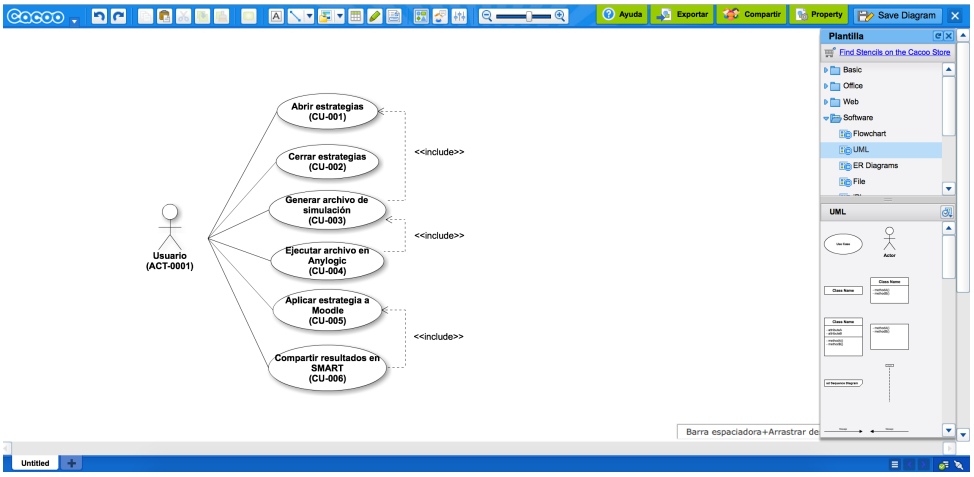

Figura 7.5: Herramienta Cacoo

7.3.6. SoapUI

SoapUI es una herramienta, desarrollada en Java, para la realización de pruebas a aplicaciones con arquitectura orientada a servicios SOA o REST. Soporta múltiples protocolos como SOAP, REST, HTTP, JMS, AMF y JDBC. Posee una versión de Código abierto y otra versión de pago realizada por la compañía SmartBear.

7.3.7. LaTeX

LATEX es un sistema de composición de textos, orientado a la creación de libros y documentos científicos. Está formado por un gran conjunto de marcos de TEX, escrito por Leslie Lamport en 1984, con la intención de facilitar el lenguaje de composición tipográfica, TEX, creado por Donald Knuth. Cuenta con la ventaja añadida de que se puede aumentar sus capacidades usando comandos propios de TEX, descritos en The TeXbook. Esto hace de LATEX una herramienta práctica y útil, ya que a su facilidad de uso se le une toda la potencia de TEX.

7.4. Mapeo Implementación-Iteraciones

En el cuadro 7.1 se muestra el mapeo de los componentes desarrollados en cada iteración.

Iteración	**Clases Implementadas**
Iteración 0	Se desarrollo el modelo de simulación base.
Iteración 1	Actividad.java, Coordenada.java, Criterio.java, Estrategia.java, Premio.java, PremioWS.java, MainApp.java
Iteración 2	StrategyOverviewController.java, MenuBarController.java, StrategyOverview.fxml, RootLayout.fxml, InfoDialog.fxml, AnylogicUtils.java, AgentUtils.java, SimulationUtils.java, MainUtils.java
Iteración 3	AboutController.java, ErrorController.java, InfoController.java, StrategyOverview.fxml, ErrorDialog.fxml, AboutView.fxml, DataMoodleUtils.java, Config.java, WSUtils.java

Cuadro 7.1: Mapeo Implementación-Iteraciones

Capítulo 8

Pruebas

En este capítulo se presenta el plan de pruebas del sistema, incluyendo los diferentes tipos de pruebas que se han llevado a cabo.

8.1. Estrategia

Todas las pruebas que se han realizado han sido de tipo manual ya que debido las características del proyecto realizar una automatización de pruebas hubiese implicado un aumento considerable del coste temporal.

Se han realizado pruebas no funcionales, pruebas unitarias y, pruebas de aceptación, donde los mismos clientes han evaluado los resultados. Para las pruebas de regresión fue necesario detectar las funciones dependientes de otras. De forma que cuando se modificase alguna que tuviese algo que ver con otra, se volvían a realizar las pruebas de la segunda.

8.1.1. Descripción del entorno de pruebas

Las pruebas se han realizado en dos entornos diferentes, para así probar la compatibilidad del sistema en distintos sistemas operativos. A continuación, se describen las especificaciones técnicas de los equipos utilizados para las pruebas.

- MacBook Pro
  - Sistema Operativo: OS X El Capitan
  - Procesador: 2,7 GHz Intel Core i5
  - Memoria: 8 GB 1867 MHz DDR3
  - Almacenamiento: 256 GB
- Portátil Dell Vostro 3558

  - Sistema Operativo: Windows 7 64 bits
  - Procesador: 2,20 GHz Intel Core i5-5200
  - Memoria: 16 GB
  - Almacenamiento: 500 GB.
- Software adicional
  - Anylogic versión PLE para Windows 7
  - Anylogic versión PLE para Mac OS
  - Instalación en servidor local de plataforma Moodle versión 3.0 con PHP 5.4.4 y MySQL 5.5.31

8.2. Pruebas no funcionales

En este apartado se describen las pruebas realizadas para validar el cumplimiento de los requisitos no funcionales.

A continuación, en los Cuadros 8.1, 8.2, 8.3, 8.4, 8.5 y 8.6, se describen las distintas pruebas no funcionales que se han realizado.

Código	PNF-001
Nombre	Prueba de entorno
Descripción	Se comprueba que la aplicación funciona correctamente en sistemas Windows y en sistemas MAC OS.

Cuadro 8.1: Pruebas no funcionales: PNF-001

Código	PNF-002
Nombre	Idioma
Descripción	Se comprueba que el idioma de la aplicación esta es el inglés.

Cuadro 8.2: Pruebas no funcionales: PNF-002

Código	PNF-003
Nombre	Usabilidad
Descripción	Se ha probado con personas sin conocimientos informáticos, teniendo resultados favorables.

Cuadro 8.3: Pruebas no funcionales: PNF-003

Código	PNF-004
Nombre	Compatibilidad
Descripción	Se comprueba que la aplicación es compatible con el software externo Anylogic.

Cuadro 8.4: Pruebas no funcionales: PNF-004

Código	PNF-005
Nombre	Rendimiento
Descripción	Se ha realizado mediciones de las tareas más críticas de la aplicación, como son la generación de ALP, las consultas a Moodle y los servicios Rest.

Cuadro 8.5: Pruebas no funcionales: PNF-005

Código	PNF-006
Nombre	Mantenibilidad
Descripción	Se ha desarrollado un sistema en el que cada funcionalidad está en un módulo diferente. Lo que favorece la capacidad de mantenimiento.

Cuadro 8.6: Pruebas no funcionales: PNF-006

8.3. Pruebas funcionales

En este apartado se describen los distintos tipos de pruebas funcionales realizados para la validación del sistema. Estas pruebas se han enfocado de manera que el sistema cumpla con todos los requisitos funcionales establecidos en el análisis del sistema.

8.3.1. Pruebas unitarias

Se han descompuesto las distintas funcionalidades del sistema, en funcionalidades mas pequeñas de las cuales se han realizado las distintas pruebas unitarias. Esto ha permitido un gran control de la aplicación ante los errores que han ido surgiendo, permitiendo una rápida localización y resolución.

A continuación, en los Cuadros 8.7, 8.8, 8.9, 8.10, 8.11, 8.12, 8.13 y 8.14, se describen las distintas pruebas unitarias que se han realizado.

Código	PU-001
Nombre	Comportamiento del modelo de simulación
Descripción	Se comprueba que el modelo de simulación base creado en Anylogic se comporta de la manera esperada, dependiendo de los parámetros introducidos.

Cuadro 8.7: Pruebas Unitarias: PU-001

Código	PU-002
Nombre	Lectura de las estrategias
Descripción	Se comprueba que la aplicación lee el archivo XML que contiene las estrategias de manera correcta.

Cuadro 8.8: Pruebas Unitarias: PU-002

Código	PU-003
Nombre	Validación del formulario de la pantalla de simulación
Descripción	Se comprueba que el formulario valida perfectamente los datos permitidos y los no permitidos.

Cuadro 8.9: Pruebas Unitarias: PU-003

Código	PU-004
Nombre	Lectura del formulario
Descripción	Se comprueba que la aplicación recibe y trata de manera correcta los datos introducidos por el usuario en el formulario de la pantalla de simulación.

Cuadro 8.10: Pruebas Unitarias: PU-004

Código	PU-005
Nombre	Escritura de archivo ALP
Descripción	Se comprueba que la aplicación realiza la escritura del archivo ALP de manera correcta.

Cuadro 8.11: Pruebas Unitarias: PU-005

Código	PU-006
Nombre	Comunicación con base de datos de Moodle
Descripción	Se comprueba que la aplicación realiza una conexión correcta con la base de datos de Moodle, así como las consultas necesarias.

Cuadro 8.12: Pruebas Unitarias: PU-006

Código	PU-007
Nombre	Tratamiento de datos de Moodle
Descripción	Se comprueba que la aplicación interpreta los datos obtenidos de Moodle de manera correcta, para la comprobación de resolución de tareas y asignaciones de recompensas.

Cuadro 8.13: Pruebas Unitarias: PU-007

Código	PU-008
Nombre	Funcionamiento de servicios web
Descripción	Se comprueba que la aplicación realiza de manera correcta la conexión con SmartWeb por medio de servicios web REST.

Cuadro 8.14: Pruebas Unitarias: PU-008

8.3.2. Pruebas de integración

Se han realizado pruebas de integración para comprobar el correcto funcionamiento de los módulos al unir las distintas funcionalidades, probadas previamente por separado. Se han realizado 2 pruebas de integración, una para el módulo de simulación, y otra para el módulo de Moodle. A continuación, se describen en los Cuadros 8.15 y 8.16.

Código	PI-001
Nombre	Funcionamiento del módulo de simulación
Descripción	Se comprueba que la aplicación realiza la funcionalidad de generar el archivo de simulación y su posterior ejecución en la aplicación Anylogic.
Comentarios	Esta prueba engloba las siguientes pruebas unitarias: PU-001, PU-002, PU-003, PU-004 y PU-005

Cuadro 8.15: Pruebas Integración: PI-001

Código	PI-002
Nombre	Funcionamiento del módulo de simulación
Descripción	Se comprueba que la aplicación realiza la funcionalidad de aplicar las estrategias a Moodle y la posterior publicación de los resultados en la red social SmartWeb.
Comentarios	Esta prueba engloba las siguientes pruebas unitarias: PU-006, PU-007 y PU-008

Cuadro 8.16: Pruebas Integración: PI-002

8.3.3. Pruebas de implantación

Una vez que la aplicación ha sido finalizada, se ha realizado la instalación de la aplicación en equipos cuyo entorno podría ser el real para la explotación de la aplicación. Estas pruebas se han realizado en un equipo con sistema operativo Windows y en un equipo con sistema operativo Mac OS. Para ello se ha realizado una prueba de sistema en cada entorno [Cuadro 8.17].

Código	PS-001
Nombre	Funcionamiento del sistema
Descripción	Se comprueba que la aplicación funciona en su totalidad de manera correcta en el entorno instalado.

Cuadro 8.17: Pruebas Sistema: PI-002

8.4. Resultado de las pruebas

Como es normal, durante la realización del plan de pruebas se han encontrado errores. Una vez detectado el error se ha realizado un análisis de este y su posterior corrección. Una vez corregido el error se han realizado pruebas de regresión para comprobar que la corrección de este no ha afectado a funcionalidades que funcionaban ya de manera correcta.
Éstas pruebas de regresión son las pruebas descritas en el apartado anterior.
A continuación en la [Figura 8.1] se describe la cantidad de errores encontrados por cada prueba unitaria. En él se puede observar que la tarea crítica de este proyecto ha

sido la generación de archivo ALP, pues es la que mas errores ha reportado durante las pruebas.

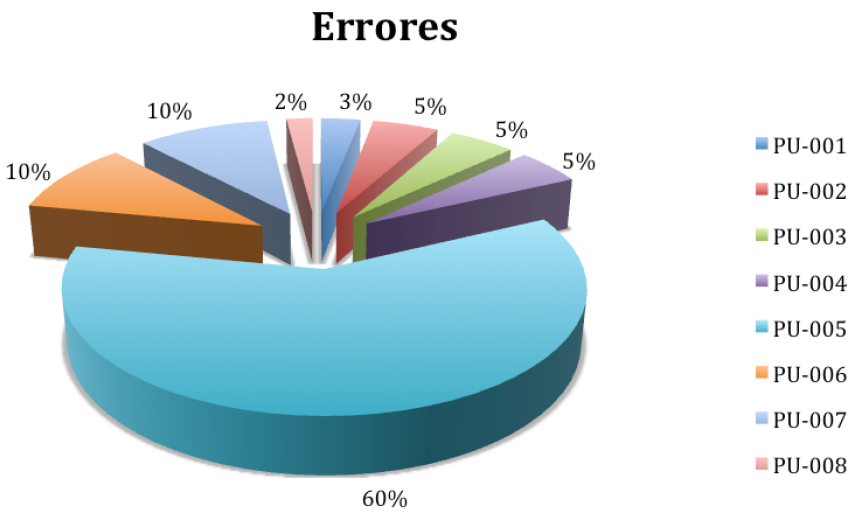

Figura 8.1: Errores encontrados en Pruebas Unitarias

8.5. Mapeo Pruebas-Iteraciones

En el cuadro 8.18 se muestra el mapeo de las pruebas realizadas en cada iteración.

Iteración	**Pruebas**
Iteración 0	PU-001
Iteración 1	PU-002
Iteración 2	PU-003, PU-004, PU-005 y PI-001
Iteración 3	PU-006, PU-007, PU-008, PI-002, PS-001, PNF-001, PNF-002, PNF-003, PNF-004, PNF-005 y PNF-006

Cuadro 8.18: Mapeo Pruebas-Iteraciones

Capítulo 9

Manuales de usuario

9.1. Descarga e Instalación de Anylogic

Para descargar Anylogic accedemos a http://www.anylogic.com/downloads.

Figura 9.1: Descargar Anylogic

Es un requisito imprescindible descargar la versión PLE. Descargaremos la versión para Windows o para Mac dependiendo del sistema operativo de nuestro equipo [Figura 9.1].

Antes de iniciar la descarga se abre una pantalla con un formulario, que tendremos que rellenar. Una vez completado el formulario podremos iniciar la descarga.

9.1.1. Instalación en Windows

Una vez descargado el archivo, que tendrá extensión .exe, haremos doble click sobre él para abrirlo.

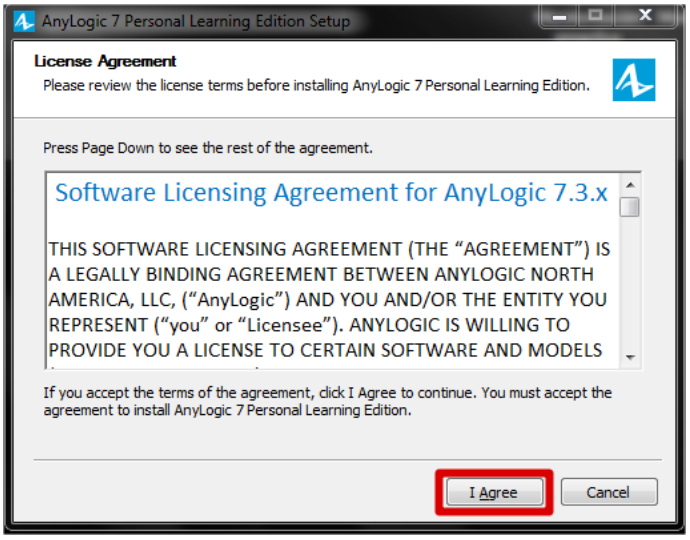

Figura 9.2: Instalación Anylogic en Windows

Aceptamos los acuerdos de licencia [Figura 9.2].

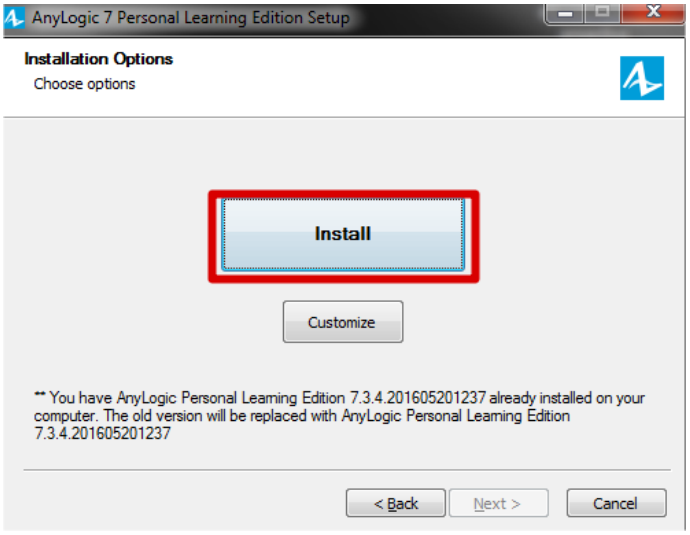

Figura 9.3: Instalación de Anylogic en Windows

Por último pulsamos el botón *Install* [Figura 9.3] y se procederá a instalarse. Una vez terminada la instalación ya tenemos el programa disponible en nuestro equipo.

9.1.2. Instalación de Anylogic en Mac

Una vez descargado el archivo, que tendrá extensión .dmg, haremos doble click sobre él para abrirlo. Se nos abrirá la siguiente pantalla.

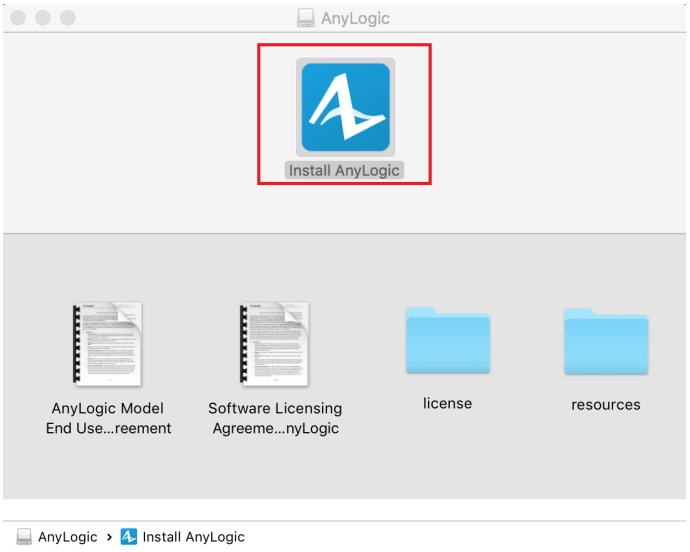

Figura 9.4: Instalación de Anylogic en Mac

Hacemos doble click sobre el icono de Anylogic [Figura 9.4]. Se nos abrirá el siguiente dialogo.

Figura 9.5: Instalación de Anylogic en Mac

Pulsamos sobre el botón *Continuar* [Figura 9.5].

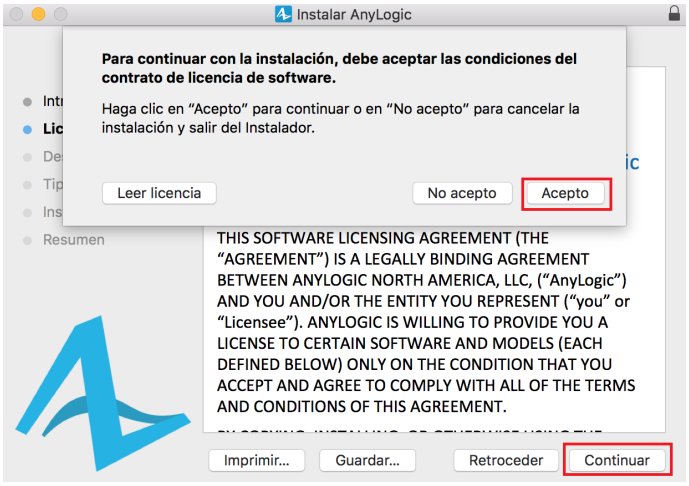

Figura 9.6: Instalación de Anylogic en Mac

Pulsamos sobre el botón *Continuar*, y se abrirá un dialogo. Pulsamos el botón *Acepto* [Figura 9.6].

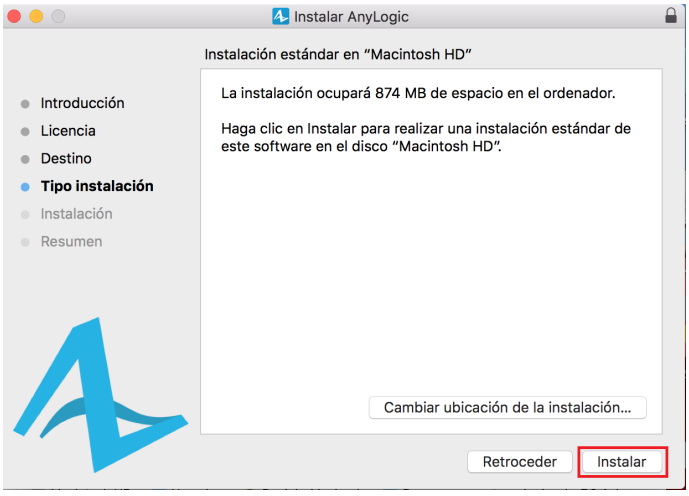

Figura 9.7: Instalación de Anylogic en Mac

Por último pulsamos el botón *Instalar* [Figura 9.7] y se procederá a instalarse. Una vez terminada la instalación ya tenemos el programa disponible en nuestro equipo.

9.2. Manual de uso de Strategy Simulator

Esta aplicación no requiere de instalación por lo que solo tenemos que abrir el archivo ejecutable. Si por defecto, se abre con algún programa que tengamos instalado en el equipo, haremos click sobre él con el botón derecho, le damos a *abrir con* y seleccionamos la opción *Java(TM) Platform SE binary*

Generar archivo de simulación

Al abrir la aplicación se muestra la pantalla principal. Para empezar a usarla lo primero que debemos hacer es cargar el archivo XML que contiene las estrategias de gamificación. Para ello pulsaremos sobre el botón *Open*, que abrirá un dialogo para que seleccionemos el archivo XML [Figura 9.8].

Figura 9.8: Uso de Strategy Simulator

Una vez cargadas las estrategias accedemos a la pantalla de simulación, para ello pulsamos el botón *Simulation* [Figura 9.9].

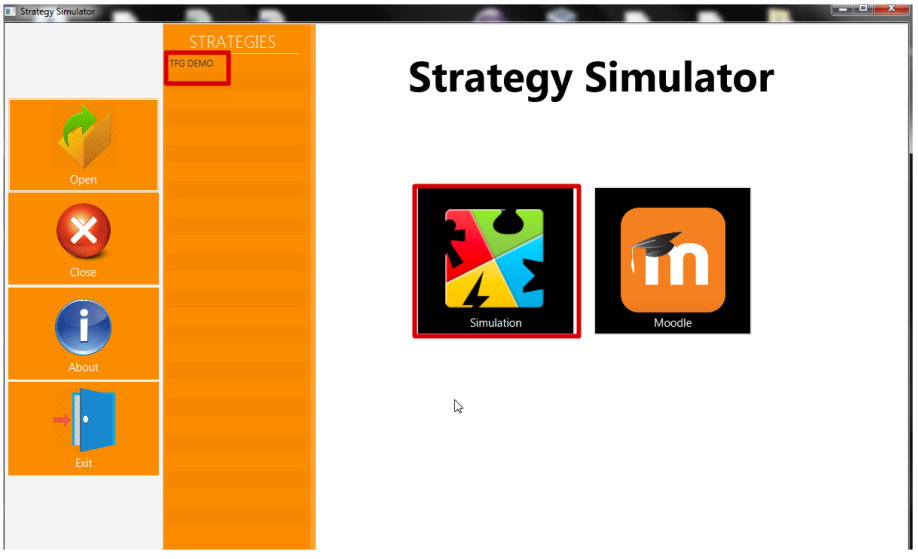

Figura 9.9: Uso de Strategy Simulator

Una vez en esta pantalla tendremos que seleccionar la estrategia con la que queremos realizar la simulación. Una vez seleccionada rellenaremos TODOS los campos del formulario, en el caso de no rellenar algún campo la aplicación nos lo indicaría con un mensaje de error. Al pulsar en el botón *Browser*, se abrirá un dialogo para que seleccionemos el Excel que contiene la tasa de éxito. Una vez que tenemos todos los

campos completados procedemos a generar el archivo de simulación pulsando sobre el botón *Generate* [Figura 9.10].

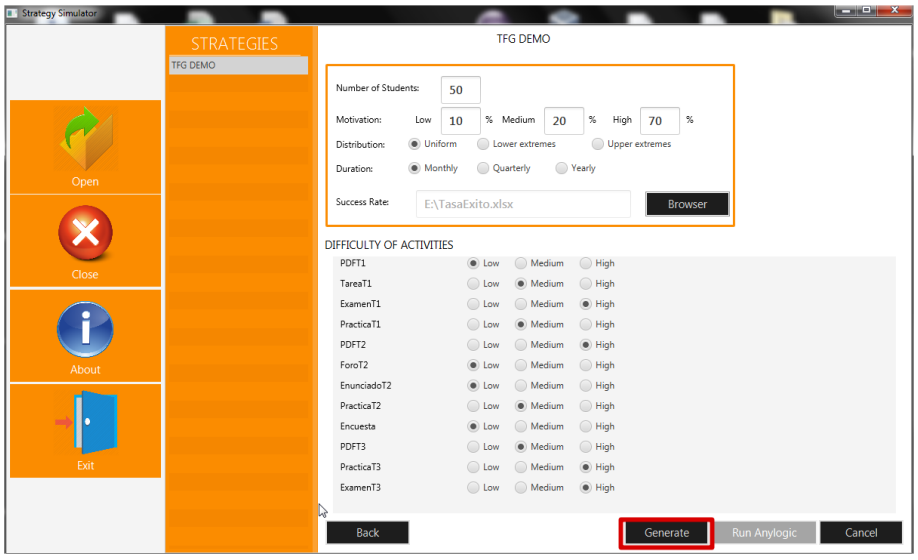

Figura 9.10: Uso de Strategy Simulator

Al pulsar sobre el botón *Generate* se abre un dialogo para que el usuario seleccione dónde quiere guardar el archivo. Una vez hecho esto se procede a generar el archivo. Tanto si la acción se ha realizado con éxito, como si no, la aplicación lo indicará mediante un mensaje por pantalla.

Una vez que tenemos el archivo de simulación generado pulsamos sobre el botón *Run Anylogic* [Figura 9.11].

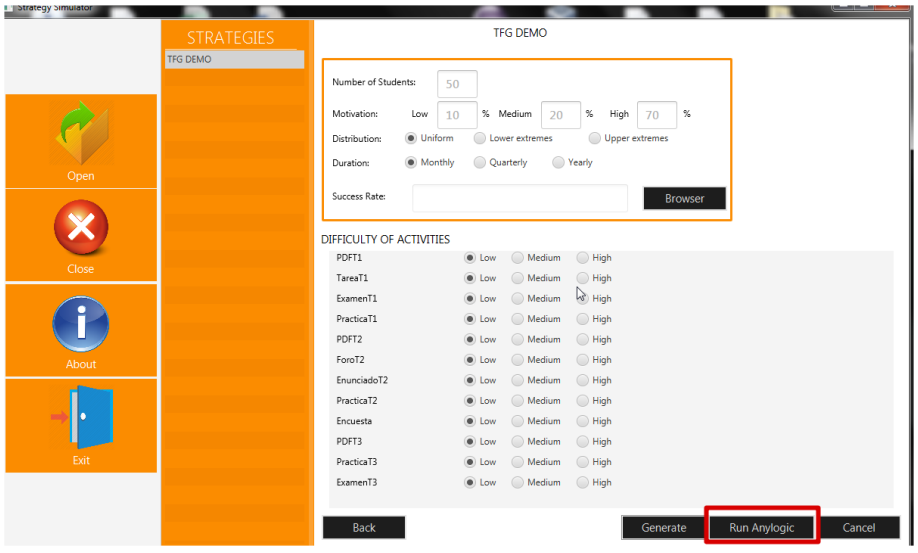

Figura 9.11: Uso de Strategy Simulator

Al pulsar este botón se abre el archivo de simulación generado en la aplicación Anylogic. Si estamos utilizando la aplicación en MAC OS, debido a un bug de Anylogic, tendremos que abrir el archivo de manera manual [Figura 9.12].

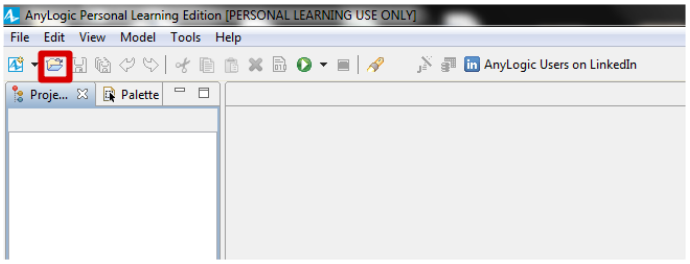

Figura 9.12: Uso de Strategy Simulator

Una vez con el archivo abierto en Anylogic procedemos a ejecutarlo pulsando el botón *Run* [Figura 9.13].

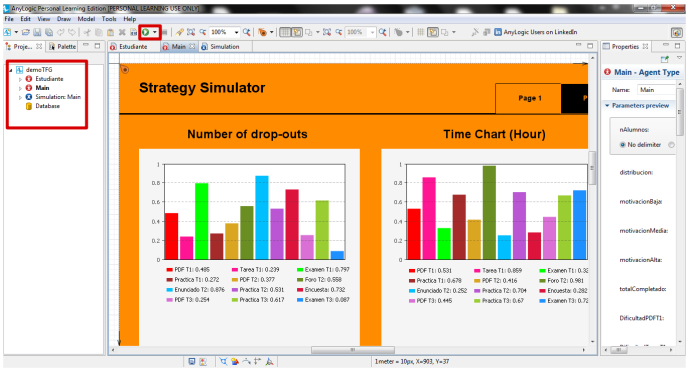

Figura 9.13: Uso de Strategy Simulator

Una vez ejecutado, se abre la pantalla principal de la simulación. En ella volvemos a tener el formulario que teníamos en Strategy Simulator [Figura 9.10]. Este formulario, está destinado a que si el usuario decide cambiar alguno de los parámetros introducidos no tenga que volver a generar el archivo de simulación. Para empezar la simulación pulsamos el botón *Run* [Figura 9.14].

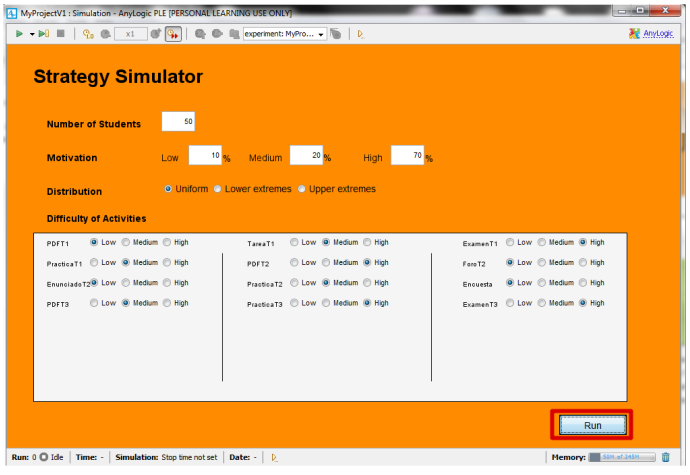

Figura 9.14: Uso de Strategy Simulator

Una vez realizada la simulación tenemos dos pantallas con los resultados obtenidos.

En la primera tenemos dos gráficas, una nos muestra los abandonos producidos en cada actividad, y otra el tiempo promedio que se ha empleado en realizar cada actividad. Para navegar a la otra pantalla pulsamos el botón *Page 2* [Figura 9.15].

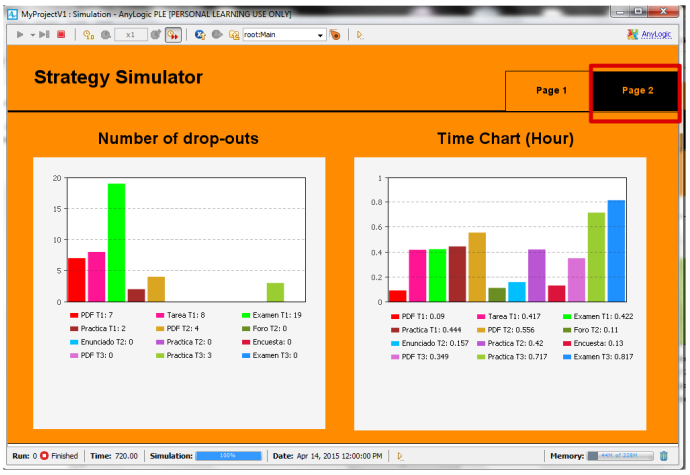

Figura 9.15: Uso de Strategy Simulator

En la segunda tenemos dos gráficas, una nos muestra las recompensas obtenidas por los alumnos del curso, y otra una clasificación del número de alumnos que han conseguido llegar a cada nivel. También muestra el número de alumnos que han completado la estrategia con éxito. Para navegar a la pantalla anterior pulsamos el botón *Page 1* [Figura 9.16].

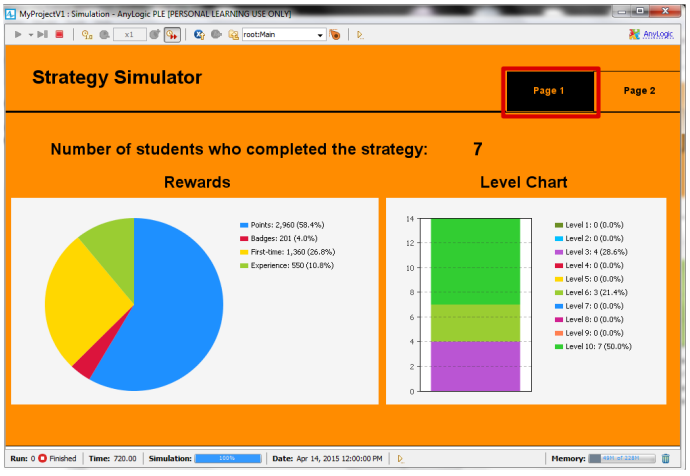

Figura 9.16: Uso de Strategy Simulator

Aplicar estrategia a Moodle

Una vez que tenemos las estrategias cargadas en la aplicación, accedemos a la pantalla de Moodle pulsando el botón *Moodle* [Figura 9.17].

Figura 9.17: Uso de Strategy Simulator

Una vez que estamos en esta pantalla, seleccionamos la estrategia que queremos aplicar y pulsamos el botón *Apply to Moodle* [Figura 9.18].

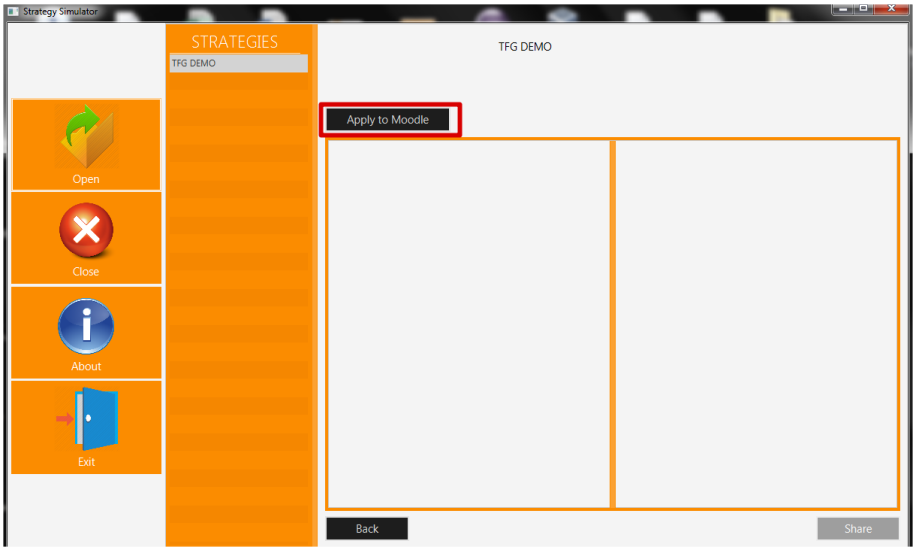

Figura 9.18: Uso de Strategy Simulator

Una vez que aplicamos la estrategia a Moodle, aparecerán dos gráficas con los resultados obtenidos. Una nos muestra las recompensas obtenidas por los alumnos del curso, y otra una clasificación del número de alumnos que han conseguido llegar a cada nivel. Por último para publicar estos resultados en la red social SmartWeb pulsamos el botón *Share* [Figura 9.19].

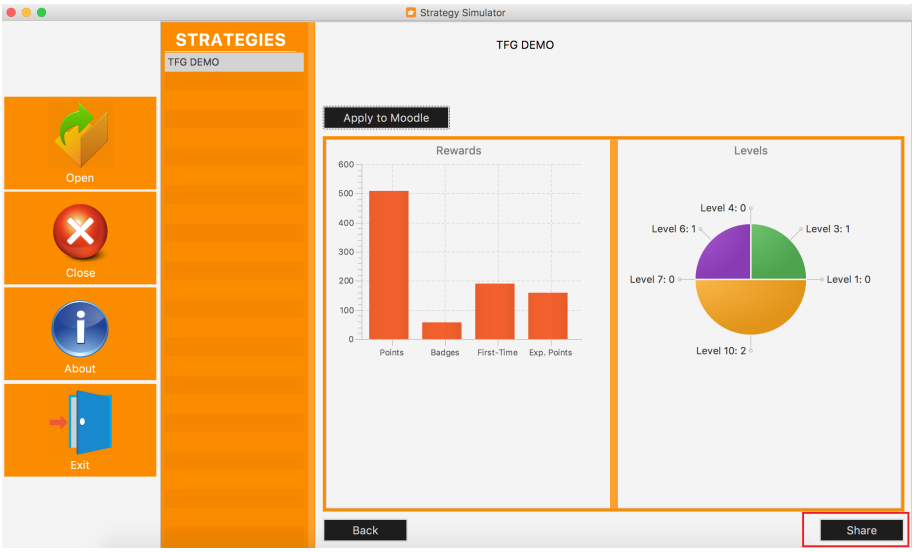

Figura 9.19: Uso de Strategy Simulator

Capítulo 10

Conclusiones

10.1. Objetivos alcanzados

Tras la finalización de este proyecto puedo decir que se ha alcanzado el principal objetivo que tenia esta aplicación.

Se ha conseguido ofrecer al usuario una herramienta capaz de simular el comportamiento de un grupo de alumnos en un curso de Moodle ante una estrategia de gamificación. Además de estos resultados de realizar la simulación se le ofrece la posibilidad de aplicar la estrategia a un curso real de Moodle y poder ver resultados, hacer comparativas y un análisis.

Al final es el objetivo global de esta aplicación que el usuario pueda realizar un análisis a partir de unos resultados fiables para posteriormente pueda realizar una toma una decisión.

10.2. Lecciones aprendidas

A nivel personal, la lección mas valiosa de este proyecto ha sido la confianza obtenida para realizar un proyecto al completo, pasando por cada una de las fases del ciclo de vida del software.

Durante la carrera no he tenido la oportunidad de afrontar un proyecto de estas características, por los que desconocía la cantidad de dificultades que van surgiendo durante el desarrollo.

Otra de las lecciones aprendidas en este proyecto es la importancia de una buena planificación, en mi caso al no trabajar de manera regular en este proyecto ha sido de vital importancia tener en todo momento un control de las tareas. Utilizar la herramienta *Trello* ha sido muy importante ya que me ha ayudado mucho a planificar estas tareas y a conocer el estado en el que se encuentran después de pasar un tiempo sin poder trabajar en el proyecto.

Para abordar todo el tema de la simulación, ha sido fundamental una investigación y estudio previo sobre el tema. Al principio pensaba estar perdiendo el tiempo ya que sientes la sensación de no avanzar en el proyecto, pero después todo este periodo de aprendizaje me ha ayudado en tener claro muchos conceptos que después he usado en la práctica.

Tras la finalización de este proyecto me he dado cuenta de mi capacidad para afrontar y solucionar problemas. Al inicio de este trabajo tenía serias dudas sobre si podría alcanzar los objetivos marcados, pero a medida que empiezas te vas dando cuenta que tienes recursos suficientes para salir adelante y finalmente lo consigues.

Quizá compaginar este proyecto con la actividad laboral aparte de retrasar el desarrollo del proyecto, me ha ayudado a la hora de tomar decisiones relacionadas con el desarrollo, el uso de herramientas, la organización del trabajo y a alcanzar cierta madurez en los conocimientos adquiridos durante la carrera.

10.3. Trabajos futuros

En esta sección, se presentan algunas mejoras que pueden ser incluidas en futuras versiones del software.

- Una de las mejoras a estudiar para versiones futuras es la lectura de las estrategias de gamificación. Actualmente estas estrategias están almacenadas en un archivo con formato XML, puede ser una buena idea que en un futuro estas estrategias estén almacenadas en una base de datos, para ello habría que modificar la forma de cargar las estrategias en la aplicación.
- En la pantalla de Moodle, al aplicar una estrategia se muestran los resultados en dos gráficas. Estas gráficas muestran los resultados obtenidos de manera muy genérica. Una mejora para el usuario sería añadir otras gráficas que pudiesen mostrar mas detalle de estos resultados, un ejemplo sería tener una visualización de qué recompensas ha conseguido cada alumno.
- Otros de los aspectos en los que se puede trabajar en un futuro es en la mejora del modelo de simulación. Actualmente, los alumnos tienen un comportamiento individual, es decir, se comportan de acuerdo a unas reglas fijas. Una mejora del modelo sería que los alumnos pudiesen interactuar entre ellos y que su comportamiento pudiese variar durante la simulación.

Bibliografía

[1] Anylogic [acceso online: 22/09/2016]: http://www.anylogic.com/

[2] Ayuda de Anylogic [acceso online: 10/05/2016]: http://help.anylogic.com/index.jsp?topic=/com.xj.anylogic.help/html/presentation/Slider.html

[3] Bloc de Anylogic [acceso online: 15/12/2015]: https://xavierpuig.wordpress.com/page/3/

[4] Website Runthemodel [acceso online: 05/05/2015]: http://www.runthemodel.com/

[5] Website de Acámica [acceso online: 10/05/2015]: https://www.acamica.com/

[6] Grigoriev, I. (2015). *Anylogic 7 in three days.*

[7] Moodle Database Schema [acceso online: 10/08/2016]: https://docs.moodle.org/dev/Database_schema_introduction

[8] Moodle ER Diagram [acceso online: 10/08/2016]: http://www.examulator.com/er/

[9] Moodle [acceso online: 10/06/2016]: https://download.moodle.org/macosx/

[10] Moodle data manipulation API [acceso online: 10/08/2016]: https://docs.moodle.org/dev/Data_manipulation_API

[11] Moodle access API [acceso online: 10/08/2016]: https://docs.moodle.org/dev/Access_API

[12] Google material design [acceso online: 05/02/2016]: https://material.

google.com/style/color.html#color-color-schemes

[13] Herramienta Trello [acceso online: 27/09/2016]: https://trello.com

[14] Navegación y diálogos en JavaFx [acceso online: 10/02/2016]: http://www.betacoders.org/aprender-javafx-parte-3-navegacion-y-dialogos/

[15] API JavaFx [acceso online: 20/08/2016]: https://docs.oracle.com/javafx/2/api/

[16] Recursos de JavaFx [acceso online: 19/11/2015]: http://javainutil.blogspot.com.es/2013/02/javafx-abrir-y-guardar-archivos.html

[17] Componentes Interfaz JavaFx [acceso online: 06/08/2016]: http://download.oracle.com/otndocs/products/javafx/2/samples/Ensemble/index.html#SAMPLES

[18] Tutorial JavaFx [acceso online: 18/09/2016]: http://code.makery.ch/library/javafx-8-tutorial/es/

[19] Bloc Mkyong [acceso online: 05/08/2016]: http://www.mkyong.com/tutorials/jax-rs-tutorials/

[20] RESTful Java client with Jersey [acceso online: 05/08/2016]: http://www.mkyong.com/webservices/jax-rs/restful-java-client-with-jersey-client/

[21] Recurso de JDBC [acceso online: 25/07/2016]: http://www.hermosaprogramacion.com/2014/07/mysql-java-conectar-como/

[22] Tabla salarial empresas TIC [acceso online: 24/09/2016]: http://www.ccoo-servicios.es/archivos/actualizaciontablas\%20XVIconvcol.pdf